
MPC5777C Reference Manual

Document Number: MPC5777CRM
Rev. 8, 11/2016





Contents

Section number	Title	Page
Chapter 1		
Preface		
1.1	Overview.....	85
1.2	Device versions.....	85
1.3	Audience.....	85
1.4	Document organization.....	86
1.5	Conventions.....	86
1.5.1	Numbering systems.....	86
1.5.2	Typographic notation.....	87
1.5.3	Special terms.....	87
1.6	References.....	88
Chapter 2		
Introduction		
2.1	Overview.....	89
2.2	Device Description.....	89
2.2.1	Family comparison.....	89
2.2.2	Block diagram.....	92
2.2.3	Packages.....	93
2.2.4	Features summary.....	93
Chapter 3		
Signal Description		
3.1	Production packages.....	97
3.2	Emulation packages.....	97
3.3	Pad ring and ball maps.....	97
3.4	Pin function selection.....	97
3.4.1	Pad output multiplexing.....	98
3.4.2	Pin input multiplexing.....	98
3.4.3	LVDS signal selection.....	98

Section number	Title	Page
3.5	External signal descriptions, multiplexing, and attributes.....	100
3.5.1	IBE and OBE control special cases.....	100
3.5.2	FCCU signals.....	109

Chapter 4 Memory Map, Interrupts, and DMA Requests

4.1	Memory map.....	111
4.2	Interrupt vector assignments.....	118
4.3	DMA request mapping.....	140

Chapter 5 Device Configuration Format (DCF) Records

5.1	Overview.....	145
5.2	DCF records in boot sequence.....	146
5.3	DCF records location.....	146
5.4	DCF record structure.....	146
5.5	DCF records sequence.....	147
5.6	DCF client Chip Select module assignments.....	149
5.7	DCF clients.....	150
5.7.1	UTEST Miscellaneous DCF client.....	157
5.7.2	UTEST Secure Boot Configurations DCF client.....	160
5.7.3	UTEST Secure Boot Code Address DCF client.....	161
5.7.4	UTEST Secure Boot Code Length DCF client.....	162
5.7.5	UTEST Analog External Oscillator (XOSC ANA) DCF client.....	163
5.7.6	UTEST Error Functions Pads Configuration DCF client.....	163
5.7.7	UTEST Reset Escalation Configuration DCF client.....	165
5.7.8	UTEST PMC Reset Enable Control (REE CTRL) DCF client.....	166
5.7.9	UTEST PMC Temperature Sensor Reset Event Enable Control (REE TMPSNS) DCF client.....	166
5.7.10	UTEST PMC Voltage Regulator Enable (VREG ENB) DCF client.....	167

Chapter 6 e200z7 Core Complex Overview

6.1	Core complex.....	169

Section number	Title	Page
6.2	Core0 and Core1 overview.....	169
6.3	Checker core overview.....	169

Chapter 7 Platform Coherency Unit (PCU)

7.1	Introduction.....	171
7.2	Memory map and register descriptions.....	171
7.2.1	PCU Configuration and Error Status Register (PCU_CESR).....	172
7.2.2	PCU Error Address Register, Core n (PCU_EAR _n).....	176
7.2.3	PCU Error Detail Register, Core n (PCU_EDR _n).....	177
7.2.4	PCU Interrupt Register, Core n (PCU_IR _n).....	178

Chapter 8 System Integration Unit (SIU, SIU_B)

8.1	Introduction.....	181
8.1.1	Block diagram.....	181
8.1.2	Overview.....	182
8.1.3	Modes of operation.....	183
8.2	Memory Map and Register Definition.....	184
8.2.1	MCU Identification Register (SIU_MIDR).....	236
8.2.2	Reset Status Register (SIU_RSR).....	237
8.2.3	System Reset Control Register (SIU_SRCR).....	240
8.2.4	External IRQ Status Register (SIU_EISR).....	241
8.2.5	DMA/Interrupt Request Enable Register (SIU_DIRER).....	244
8.2.6	DMA/Interrupt Request Select Register (SIU_DIRSR).....	247
8.2.7	Overrun Status Register (SIU_OSR).....	249
8.2.8	Overrun Request Enable Register (SIU_ORER).....	251
8.2.9	IRQ Rising-Edge Event Enable Register (SIU_IREER).....	253
8.2.10	IRQ Falling-Edge Event Enable Register (SIU_IFEER).....	255
8.2.11	IRQ Digital Filter Register (SIU_IDFR).....	258
8.2.12	IRQ Filtered Input Register (SIU_IFIR).....	258

Section number	Title	Page
8.2.13	Pad Configuration Register (SIU_PCR n).....	260
8.2.14	GPIO Pin Data Output Register (SIU_GPDO n).....	263
8.2.15	GPIO Pin Data Input Register (legacy) (SIU_GPDInL).....	264
8.2.16	External IRQ Input Select Register (SIU_EIISR).....	265
8.2.17	DSPI Input Select Register (SIU_DISR).....	267
8.2.18	eQADC Command FIFO Trigger Source Registers (SIU_ISEL4).....	269
8.2.19	eQADC Command FIFO Trigger Source Registers (SIU_ISEL5).....	274
8.2.20	eQADC Command FIFO Trigger Source Registers (SIU_ISEL6).....	277
8.2.21	eQADC Command FIFO Trigger Source Registers (SIU_ISEL7).....	281
8.2.22	eTPU_A Input Select Register (SIU_ISEL8).....	284
8.2.23	eQADC Advance Trigger Source Register (SIU_ISEL9).....	285
8.2.24	Decimation Filter Register 1 (SIU_DECFIL1).....	288
8.2.25	Decimation Filter Register 2 (SIU_DECFIL2).....	290
8.2.26	Decimation Filter Register 3 (SIU_DECFIL3).....	293
8.2.27	Decimation Filter Register 4 (SIU_DECFIL4).....	295
8.2.28	Decimation Filter Register 5 (SIU_DECFIL5).....	297
8.2.29	Reaction Module Trigger Selection Register (SIU_REACTSR).....	298
8.2.30	Sigma-Delta Gate Selection Register (SIU_SDGATE_SEL).....	301
8.2.31	Sigma-Delta to eTPU_A Connection Configuration Register (SIU_SDETPUA).....	302
8.2.32	Sigma-Delta to eTPU_B Connection Configuration Register (SIU_SDETPUB).....	305
8.2.33	Sigma-Delta to eTPU_C Connection Configuration Register (SIU_SDETPUC).....	307
8.2.34	Trigger Burst Generator Control Registers for eQADC_A (SIU_TBG_CR_A n).....	310
8.2.35	Trigger Burst Generator Control Registers for eQADC_B (SIU_TBG_CR_B n).....	311
8.2.36	External Clock Control Register (SIU_ECCR).....	312
8.2.37	System Clock Register (SIU_SYSDIV).....	313
8.2.38	Halt Register 1 (SIU_HLT1).....	316
8.2.39	Halt Acknowledge Register 1 (SIU_HLTACK1).....	321
8.2.40	Core0 Reset Vector Register (SIU_RSTVEC0).....	324
8.2.41	Core1 Reset Vector Register (SIU_RSTVEC1).....	325

Section number	Title	Page
8.2.42	Core0 PID mapping control register (SIU_C0PID).....	327
8.2.43	Core1 PID mapping control register (SIU_C1PID).....	328
8.2.44	Halt Register 2 (SIU_HLT2).....	329
8.2.45	Halt Acknowledge Register 2 (SIU_HLTACK2).....	332
8.2.46	Sigma-Delta Clock Configuration Register (SIU_SDCLKCFG).....	334
8.2.47	LFast Clock Configuration Register (SIU_LFCLKCFG).....	335
8.2.48	PSI5 Clock Configuration Register (SIU_PSCLKCFG).....	336
8.2.49	Reset Configuration Register (SIU_RCR).....	337
8.2.50	Lock-Step Status Register (SIU_LOCKSTEP).....	337
8.2.51	Progressive Clock Switch Interrupt Enable Register (SIU_PCSEIR).....	338
8.2.52	Progressive Clock Switch Interrupt Flag Register (SIU_PCSIFR).....	339
8.2.53	Fast Ethernet Controller Control Register (SIU_FECCR).....	340
8.2.54	ECC Error Injection Register (SIU_ECCEIR).....	340
8.2.55	Parallel GPIO Pin Data Output Registers (SIU_PGPDOn).....	341
8.2.56	Parallel GPIO Pin Data Input Registers (SIU_PGPDIn).....	345
8.2.57	Masked Parallel GPIO Pin Data Output Registers (SIU_MPGPDOn).....	349
8.2.58	Mask-Output High Register (SIU_DSPIInH).....	353
8.2.59	Mask-Output Low Register (SIU_DSPIInL).....	357
8.2.60	Serialized Output Signal Selection Register for DSPI_A (SIU_ETPUBA).....	362
8.2.61	Serialized Output Signal Selection Register for DSPI_A (SIU_EMIOA).....	366
8.2.62	Serialized Output Signal Selection Register for DSPI_A (SIU_DSPIAHLA).....	369
8.2.63	Serialized Output Signal Selection Register for DSPI_B (SIU_ETPUAB).....	373
8.2.64	Serialized Output Signal Selection Register for DSPI_B (SIU_EMIOB).....	377
8.2.65	Serialized Output Signal Selection Register for DSPI_B (SIU_DSPIBHLB).....	381
8.2.66	Serialized Output Signal Selection Register for DSPI_C (SIU_ETPUAC).....	385
8.2.67	Serialized Output Signal Selection Register for DSPI_C (SIU_EMIOB).....	388
8.2.68	Serialized Output Signal Selection Register for DSPI_C (SIU_DSPICHLC).....	392
8.2.69	Serialized Output Signal Selection Register for DSPI_C (SIU_ETPUBC).....	396
8.2.70	Serialized Output Signal Selection Register for DSPI_D (SIU_ETPUBD).....	400

Section number	Title	Page
8.2.71	Serialized Output Signal Selection Register for DSPI_D (SIU_EMIOSD).....	402
8.2.72	Serialized Output Signal Selection Register for DSPI_D (SIU_DSPIDHLD).....	403
8.2.73	GPIO Pin Data Input Register (SIU_GPDIn).....	407
8.2.74	Input Multiplexing Register0 (SIU_IMUX0).....	408
8.2.75	Input Multiplexing Register1 (SIU_IMUX1).....	410
8.2.76	Input Multiplexing Register2 (SIU_IMUX2).....	411
8.2.77	Input Multiplexing Register3 (SIU_IMUX3).....	413
8.2.78	Input Multiplexing Register4 (SIU_IMUX4).....	414
8.2.79	Input Multiplexing Register5 (SIU_IMUX5).....	416
8.2.80	Input Multiplexing Register7 (SIU_IMUX7).....	418
8.2.81	Input Multiplexing Register10 (SIU_IMUX10).....	420
8.2.82	Input Multiplexing Register12 (SIU_IMUX12).....	422
8.3	SIU_B.....	423
8.3.1	Pad Configuration Register Mirrored (SIU_B_PCRnM).....	424
8.3.2	GPIO Pin Data Output Register Mirrored (SIU_B_GPDOnM).....	427
8.3.3	GPIO Pin Data Input Register Mirrored (SIU_B_GPDInM).....	428
8.4	Functional Description.....	429
8.4.1	Pad configuration.....	429
8.4.2	Interrupts.....	429
8.4.3	General Purpose I/O (GPIO).....	433
8.4.4	Pin serialization for DSPI.....	433
8.4.5	Internal multiplexing.....	435
8.4.6	Trigger Burst Generators (TBG).....	436

Chapter 9 Crossbar Switch (XBAR)

9.1	Chip-specific XBAR information.....	441
9.1.1	XBAR master and slave assignments.....	441
9.1.2	Unimplemented PRSn and CRSn registers.....	442
9.1.3	Reset value of PRSn and CRSn registers.....	442

Section number	Title	Page
9.1.4	Master priority elevation.....	443
9.2	Introduction.....	445
9.2.1	Features.....	445
9.3	Memory Map / Register Definition.....	446
9.3.1	Priority Registers Slave (XBAR_PRSn).....	447
9.3.2	Control Register (XBAR_CRSn).....	450
9.4	Functional Description.....	452
9.4.1	General operation.....	452
9.4.2	Register coherency.....	453
9.4.3	Arbitration.....	453
9.5	Initialization/application information.....	455

Chapter 10

Crossbar Integrity Checker (XBIC)

10.1	Chip-specific XBIC information.....	457
10.1.1	XBIC master and slave assignments.....	457
10.1.2	Unimplemented MCR and ESR fields.....	457
10.2	Overview.....	457
10.3	Features.....	457
10.4	Block diagram.....	458
10.5	External signal description.....	458
10.6	Memory map and register definition.....	459
10.6.1	XBIC Module Control Register (XBIC_MCR).....	459
10.6.2	XBIC Error Injection Register (XBIC_EIR).....	461
10.6.3	XBIC Error Status Register (XBIC_ESR).....	462
10.6.4	XBIC Error Address Register (XBIC_EAR).....	464
10.7	Functional description.....	465

Section number	Title	Page

Chapter 11 Peripheral Bridge (AIPS/PBRIDGE)

11.1	Chip-specific AIPS/PBRIDGE information.....	467
11.1.1	Modules assigned to PACRn and OPACRn fields.....	467
11.1.2	AIPS/PBRIDGE register reset values.....	470
11.2	Introduction.....	471
11.2.1	Features.....	471
11.2.2	General operation.....	471
11.3	Memory map/register definition.....	472
11.3.1	Master Privilege Register A (AIPS_MPRA).....	472
11.3.2	Master Privilege Register B (AIPS_MPRB).....	476
11.3.3	Peripheral Access Control Register (AIPS_PACRn).....	478
11.3.4	Off-Platform Peripheral Access Control Register (AIPS_OPACRn).....	483
11.4	Functional description.....	488
11.4.1	Access support.....	488

Chapter 12 Platform Configuration Module (PCM)

12.1	PCM memory map and register descriptions.....	489
12.1.1	FEC Burst Optimization Master Control Register (PCM_FBOMCR).....	490
12.1.2	Bus Bridge Configuration Register 1 (PCM_IAHB_BE1).....	491
12.1.3	Bus Bridge Configuration Register 2 (PCM_IAHB_BE2).....	494

Chapter 13 Semaphores (SEMA4)

13.1	Introduction	497
13.1.1	Multi-Core Programming 101: Software Gates.....	497
13.1.2	Overview.....	499
13.1.3	Features.....	500
13.1.4	Modes of Operation.....	501
13.2	External Signal Description.....	501

Section number	Title	Page
13.3	Memory map and register definition.....	502
13.3.1	Semaphores Gate 0 Register (SEMA4_Gate00).....	503
13.3.2	Semaphores Gate 1 Register (SEMA4_Gate01).....	504
13.3.3	Semaphores Gate 2 Register (SEMA4_Gate02).....	505
13.3.4	Semaphores Gate 3 Register (SEMA4_Gate03).....	506
13.3.5	Semaphores Gate 4 Register (SEMA4_Gate04).....	507
13.3.6	Semaphores Gate 5 Register (SEMA4_Gate05).....	508
13.3.7	Semaphores Gate 6 Register (SEMA4_Gate06).....	509
13.3.8	Semaphores Gate 7 Register (SEMA4_Gate07).....	510
13.3.9	Semaphores Gate 8 Register (SEMA4_Gate08).....	511
13.3.10	Semaphores Gate 9 Register (SEMA4_Gate09).....	512
13.3.11	Semaphores Gate 10 Register (SEMA4_Gate10).....	513
13.3.12	Semaphores Gate 11 Register (SEMA4_Gate11).....	514
13.3.13	Semaphores Gate 12 Register (SEMA4_Gate12).....	515
13.3.14	Semaphores Gate 13 Register (SEMA4_Gate13).....	516
13.3.15	Semaphores Gate 14 Register (SEMA4_Gate14).....	517
13.3.16	Semaphores Gate 15 Register (SEMA4_Gate15).....	518
13.3.17	Semaphores Processor n IRQ Notification Enable (SEMA4_CPnINE).....	519
13.3.18	Semaphores Processor n IRQ Notification (SEMA4_CPnNTF).....	521
13.3.19	Semaphores (Secure) Reset Gate n (SEMA4_RSTGT).....	522
13.3.20	Semaphores (Secure) Reset IRQ Notification (SEMA4_RSTNTF).....	524
13.4	Functional Description.....	525
13.4.1	SEMA4_GATE _n Operation.....	525
13.4.2	SEMA4_CP _n NTF Operation.....	527
13.5	Initialization Information.....	530
13.6	Application Information.....	530

Chapter 14

Memory Protection Unit (MPU)

14.1	Chip-specific MPU information.....	533
14.1.1	MPU master and slave assignments.....	533
14.1.2	MPU register reset values.....	534
14.2	Introduction.....	534
14.3	Overview.....	534
14.3.1	Block diagram.....	534
14.3.2	Features.....	535
14.4	Memory map/register definition.....	536
14.4.1	Control/Error Status Register (MPU_CESR).....	540
14.4.2	Error Address Register, slave port n (MPU_EAR _n).....	541
14.4.3	Error Detail Register, slave port n (MPU_EDR _n).....	542
14.4.4	Region Descriptor n, Word 0 (MPU_RGD _n _WORD0).....	543
14.4.5	Region Descriptor n, Word 1 (MPU_RGD _n _WORD1).....	544
14.4.6	Region Descriptor n, Word 2 (MPU_RGD _n _WORD2).....	544
14.4.7	Region Descriptor n, Word 3 (MPU_RGD _n _WORD3).....	547
14.4.8	Region Descriptor Alternate Access Control n (MPU_RGDAAC _n).....	548
14.5	Functional description.....	550
14.5.1	Access evaluation macro.....	550
14.5.2	Putting it all together and error terminations.....	552
14.5.3	Power management.....	553
14.6	Initialization information.....	553
14.7	Application information.....	553

Chapter 15

Interrupt Controller (INTC)

15.1	Chip-specific INTC information.....	557
15.2	Introduction.....	557
15.3	Block diagram.....	558

Section number	Title	Page
15.4	Features.....	560
15.5	Modes of operation.....	560
15.5.1	Software vector mode.....	560
15.5.2	Hardware vector mode.....	563
15.6	External signal description.....	564
15.7	Memory map and register definition.....	564
15.7.1	INTC Module Configuration Register (INTC_MCR).....	579
15.7.2	INTC Current Priority Register for Processor <i>n</i> (INTC_CPR_PRC <i>n</i>).....	580
15.7.3	INTC Interrupt Acknowledge Register for Processor <i>n</i> (INTC_IACKR_PRC <i>n</i>).....	581
15.7.4	INTC End Of Interrupt Register for Processor <i>n</i> (INTC_EOIR_PRC <i>n</i>).....	582
15.7.5	INTC Software Set/Clear Interrupt Register (INTC_SSCIR <i>n</i>).....	583
15.7.6	INTC Priority Select Register (INTC_PSR <i>n</i>).....	584
15.8	Functional description.....	584
15.8.1	Interrupt request sources.....	585
15.8.2	Priority management.....	586
15.8.3	Handshaking with processor.....	588
15.9	Initialization/application information.....	591
15.9.1	Initialization flow.....	591
15.9.2	Interrupt exception handler.....	592
15.9.3	ISR, RTOS, and task hierarchy.....	593
15.9.4	Order of execution.....	594
15.9.5	Priority ceiling protocol.....	595
15.9.6	Selecting priorities according to request rates and deadlines.....	598
15.9.7	Software-settable interrupt requests.....	599
15.9.8	Lowering priority within an ISR.....	600
15.9.9	Negating an interrupt request outside of its ISR.....	600
15.9.10	Examining LIFO contents.....	601
15.10	Interrupt sources.....	602

Chapter 16

Direct Memory Access Controller (eDMA)

16.1	Chip-specific eDMA information.....	603
16.2	Introduction.....	603
16.2.1	eDMA system block diagram.....	603
16.2.2	Block parts.....	604
16.2.3	Features.....	605
16.3	Modes of operation.....	607
16.4	Memory map/register definition.....	607
16.4.1	TCD memory.....	607
16.4.2	TCD initialization.....	608
16.4.3	TCD structure.....	608
16.4.4	Reserved memory and bit fields.....	609
16.4.5	Control Register (DMA_CR).....	660
16.4.6	Error Status Register (DMA_ES).....	663
16.4.7	Enable Request Register High (DMA_ERQH).....	666
16.4.8	Enable Request Register Low (DMA_ERQL).....	669
16.4.9	Enable Error Interrupt Register High (DMA_EEIH).....	673
16.4.10	Enable Error Interrupt Register Low (DMA_EEIL).....	676
16.4.11	Set Enable Request Register (DMA_SERQ).....	680
16.4.12	Clear Enable Request Register (DMA_CERQ).....	681
16.4.13	Set Enable Error Interrupt Register (DMA_SEEI).....	682
16.4.14	Clear Enable Error Interrupt Register (DMA_CEEI).....	683
16.4.15	Clear Interrupt Request Register (DMA_CINT).....	684
16.4.16	Clear Error Register (DMA_CERR).....	685
16.4.17	Set START Bit Register (DMA_SSRT).....	686
16.4.18	Clear DONE Status Bit Register (DMA_CDNE).....	687
16.4.19	Interrupt Request Register High (DMA_INTH).....	687
16.4.20	Interrupt Request Register Low (DMA_INTL).....	691

Section number	Title	Page
16.4.21	Error Register High (DMA_ERRH).....	695
16.4.22	Error Register Low (DMA_ERRL).....	699
16.4.23	Hardware Request Status Register High (DMA_HRSH).....	702
16.4.24	Hardware Request Status Register Low (DMA_HRSL).....	706
16.4.25	Global PCU Output Register (DMA_GPORn).....	710
16.4.26	Channel n Priority Register (DMA_DCHPRIn).....	711
16.4.27	TCD Source Address (DMA_TCDn_SADDR).....	712
16.4.28	TCD Transfer Attributes (DMA_TCDn_ATTR).....	712
16.4.29	TCD Signed Source Address Offset (DMA_TCDn_SOFF).....	713
16.4.30	TCD Minor Byte Count (Minor Loop Mapping Disabled) (DMA_TCDn_NBYTES_MLNO).....	713
16.4.31	TCD Signed Minor Loop Offset (Minor Loop Mapping Enabled and Offset Disabled) (DMA_TCDn_NBYTES_MLOFFNO).....	714
16.4.32	TCD Signed Minor Loop Offset (Minor Loop Mapping and Offset Enabled) (DMA_TCDn_NBYTES_MLOFFYES).....	715
16.4.33	TCD Last Source Address Adjustment (DMA_TCDn_SLAST).....	716
16.4.34	TCD Destination Address (DMA_TCDn_DADDR).....	717
16.4.35	TCD Current Minor Loop Link, Major Loop Count (Channel Linking Enabled) (DMA_TCDn_CITER_ELINKYES).....	717
16.4.36	TCD Current Minor Loop Link, Major Loop Count (Channel Linking Disabled) (DMA_TCDn_CITER_ELINKNO).....	718
16.4.37	TCD Signed Destination Address Offset (DMA_TCDn_DOFF).....	719
16.4.38	TCD Last Destination Address Adjustment/Scatter Gather Address (DMA_TCDn_DLASTSGA).....	720
16.4.39	TCD Beginning Minor Loop Link, Major Loop Count (Channel Linking Enabled) (DMA_TCDn_BITER_ELINKYES).....	720
16.4.40	TCD Beginning Minor Loop Link, Major Loop Count (Channel Linking Disabled) (DMA_TCDn_BITER_ELINKNO).....	721
16.4.41	TCD Control and Status (DMA_TCDn_CSR).....	722
16.5	Functional description.....	724
16.5.1	eDMA basic data flow.....	724
16.5.2	Fault reporting and handling.....	727

Section number	Title	Page
16.5.3	Channel preemption.....	730
16.5.4	Performance.....	730
16.6	Initialization/application information.....	735
16.6.1	eDMA initialization.....	735
16.6.2	Programming errors.....	737
16.6.3	Arbitration mode considerations.....	738
16.6.4	Performing DMA transfers.....	740
16.6.5	Monitoring transfer descriptor status.....	744
16.6.6	Channel Linking.....	746
16.6.7	Dynamic programming.....	747

Chapter 17 Error Injection Module (EIM)

17.1	Chip-specific EIM information.....	751
17.1.1	EIM channels.....	751
17.1.2	EIM channel assignments.....	751
17.1.3	EIM_EICHDN_Word register bit mapping.....	752
17.2	Introduction.....	752
17.2.1	Overview.....	753
17.2.2	Features.....	754
17.3	Memory map and register definition.....	754
17.3.1	Error Injection Module Configuration Register (EIM_EIMCR).....	756
17.3.2	Error Injection Channel Enable register (EIM_EICHEN).....	757
17.3.3	Error Injection Channel Descriptor, Word0 (EIM_EICHDN_WORD0).....	759
17.3.4	Error Injection Channel Descriptor, Word1 (EIM_EICHDN_WORD1).....	760
17.3.5	Error Injection Channel Descriptor, Word2 (EIM_EICHDN_WORD2).....	760
17.4	Functional description.....	761

Section number	Title	Page
Chapter 18		
Error Reporting Module (ERM)		
18.1	Chip-specific ERM information.....	763
18.1.1	ERM memory channels.....	763
18.1.2	Sources of memory error events.....	763
18.1.3	Managing single-bit faults of TCD RAM for eDMA_A.....	764
18.2	Introduction.....	764
18.2.1	Overview.....	764
18.2.2	Features.....	765
18.3	Memory map and register definition.....	765
18.3.1	ERM Configuration Register (ERM_CR0).....	767
18.3.2	ERM Configuration Register (ERM_CR1).....	770
18.3.3	ERM Configuration Register (ERM_CR2).....	773
18.3.4	ERM Status Register (ERM_SR0).....	776
18.3.5	ERM Status Register (ERM_SR1).....	780
18.3.6	ERM Status Register (ERM_SR2).....	784
18.3.7	ERM Memory <i>n</i> Error Address Register (ERM_EAR _{<i>n</i>}).....	786
18.3.8	ERM Memory <i>n</i> Syndrome Register (ERM_SYN _{<i>n</i>}).....	787
18.4	Functional description.....	787
18.4.1	Single-bit correction events.....	787
18.4.2	Non-correctable error events.....	788
18.5	Initialization.....	789
Chapter 19		
Clocking		
19.1	Introduction.....	791
19.2	Clock generation.....	792
19.3	SIU registers.....	793
19.4	Operating frequencies.....	794
19.5	Default clock configuration.....	795

Section number	Title	Page
19.6	Clock sources.....	796
19.6.1	16 MHz RC oscillator (IRC).....	796
19.6.2	External oscillator (XOSC).....	796
19.6.3	PLL.....	798
19.7	Clock gating.....	800
19.8	Core and on-platform clocking.....	801
19.9	Peripheral clocks.....	802
19.9.1	eTPU clocking.....	803
19.9.2	FlexCAN and MCAN clocking.....	804
19.9.3	LFAST clocking.....	804
19.9.4	PSI5 clocking.....	805
19.9.5	Ethernet clocking.....	805
19.9.6	SDADC clocking.....	805
19.10	Serial boot clocking.....	805
19.11	MCKO.....	806
19.12	CLKOUT / ENGCLK.....	806
19.13	Clock monitoring.....	806
19.13.1	CMU-monitored clocks.....	806
19.13.2	CMU_0 configuration.....	807
19.13.3	CMU_1 to CMU_8 configuration.....	808
19.13.4	CMU output signals to FCCU and INTC.....	809
19.13.5	CMU register and field availability	810
19.13.6	CMU_0_CSR reset value.....	810
19.13.7	CMU use restrictions.....	810
19.13.8	Loss-of-clock behavior.....	811
19.14	PLL loss of lock interrupt.....	811
19.15	Debug mode.....	812

Section number	Title	Page
Chapter 20		
Progressive Clock Switch (PCS)		
20.1	Introduction.....	813
20.2	Block diagram.....	813
20.3	Features.....	813
20.4	Memory map and register definition.....	814
20.4.1	PCS Switch Duration Register (PCS_SDUR).....	815
20.4.2	PCS Divider Change Register for XOSC (PCS_DIVC1).....	815
20.4.3	PCS Divider End Register for XOSC (PCS_DIVE1).....	816
20.4.4	PCS Divider Start Register for XOSC (PCS_DIVS1).....	816
20.4.5	PCS Divider Change Register for PLL1 (PCS_DIVC2).....	817
20.4.6	PCS Divider End Register for PLL1 (PCS_DIVE2).....	818
20.4.7	PCS Divider Start Register for PLL1 (PCS_DIVS2).....	818
20.4.8	PCS Divider Change Register for PLL0 (PCS_DIVC3).....	819
20.4.9	PCS Divider End Register for PLL0 (PCS_DIVE3).....	819
20.4.10	PCS Divider Start Register for PLL0 (PCS_DIVS3).....	820
20.5	Functional description.....	821
20.5.1	Generic clock change requirements.....	822
20.5.2	Configuration of PCS_SDUR.....	822
20.5.3	Configuration of PCS_DIVCn[RATE].....	822
20.5.4	Generic clock change properties.....	823
20.5.5	Clock ramp-down.....	823
20.5.6	Clock ramp-up.....	824
Chapter 21		
Dual PLL Digital Interface (PLLDIG)		
21.1	Chip-specific PLLDIG information.....	827
21.2	Introduction.....	827
21.3	Acronyms and Abbreviations.....	827
21.4	Block Diagram.....	828

Section number	Title	Page
21.5	Features.....	828
21.6	Modes of operation.....	828
21.6.1	Normal mode with reference, PLL0 or both PLLs enabled.....	828
21.7	Memory map and register definition.....	829
21.7.1	PLLDIG PLL0 Control Register (PLLDIG_PLL0CR).....	830
21.7.2	PLLDIG PLL0 Status Register (PLLDIG_PLL0SR).....	831
21.7.3	PLLDIG PLL0 Divider Register (PLLDIG_PLL0DV).....	832
21.7.4	PLLDIG PLL1 Control Register (PLLDIG_PLL1CR).....	834
21.7.5	PLLDIG PLL1 Status Register (PLLDIG_PLL1SR).....	835
21.7.6	PLLDIG PLL1 Divider Register (PLLDIG_PLL1DV).....	836
21.7.7	PLLDIG PLL1 Frequency Modulation Register (PLLDIG_PLL1FM).....	837
21.7.8	PLLDIG PLL1 Fractional Divide Register (PLLDIG_PLL1FD).....	838
21.8	Functional description.....	839
21.8.1	Input clock frequency.....	839
21.8.2	Clock configuration.....	840
21.8.3	Loss of lock.....	842
21.8.4	Frequency modulation.....	842
21.9	Maximum lock time.....	845
21.10	Initialization information.....	845

Chapter 22

Clock Monitor Unit (CMU)

22.1	Chip-specific CMU information.....	847
22.2	Introduction.....	847
22.2.1	Main features.....	848
22.3	Block diagram.....	848
22.4	Signals.....	848
22.5	Register description and memory map.....	849
22.5.1	CMU Control Status Register (CMU_CSR).....	850
22.5.2	CMU Frequency Display Register (CMU_FDR).....	851

Section number	Title	Page
22.5.3	CMU High Frequency Reference Register CLKMN1 (CMU_HFREFR).....	852
22.5.4	CMU Low Frequency Reference Register CLKMN1 (CMU_LFREFR).....	852
22.5.5	CMU Interrupt Status Register (CMU_ISR).....	853
22.5.6	CMU Measurement Duration Register (CMU_MDR).....	854
22.6	Functional description.....	855
22.6.1	Frequency meter.....	855
22.6.2	CLKMN0_RMT supervisor.....	855
22.6.3	CLKMN1 supervisor.....	856

Chapter 23 Embedded Memories

23.1	Overview.....	859
23.2	End-to-end Error Correction Code (e2eECC).....	859

Chapter 24 RAM Controller (PRAMC)

24.1	Chip-specific PRAM information.....	863
24.1.1	PRAM optional read wait states.....	863
24.1.2	Reset effects on System RAM accesses.....	863
24.2	Introduction.....	864
24.3	SRAM controller memory map and register definition.....	866
24.3.1	Platform RAM Configuration Register 1 (PRAMC_PRCR1).....	866
24.4	Functional description.....	868
24.4.1	Read and Write operations.....	868
24.4.2	Writes.....	869
24.5	Initialization/application information.....	871
24.6	Reliability considerations.....	871
24.6.1	Hsiao ECC algorithm.....	871
24.6.2	Transaction monitor.....	874

Chapter 25 Flash Memory Controller (PFLASH)

25.1	Chip-specific PFLASH information.....	877

Section number	Title	Page
25.2	Introduction.....	877
25.3	Features.....	878
25.4	Block diagrams.....	878
25.5	Flash memory controller memory map.....	879
25.5.1	Platform Flash Configuration Register 1 (PFLASH_PFCR1).....	881
25.5.2	Platform Flash Configuration Register 2 (PFLASH_PFCR2).....	884
25.5.3	Platform Flash Configuration Register 3 (PFLASH_PFCR3).....	887
25.5.4	Platform Flash Access Protection Register (PFLASH_PFAPR).....	889
25.6	Functional description.....	892
25.6.1	Basic interface protocol.....	892
25.6.2	Access protections.....	892
25.6.3	Access pipelining.....	893
25.6.4	Line read buffers and prefetch operation.....	894
25.6.5	Instruction/Data prefetch triggering.....	895
25.6.6	Per-Master prefetch triggering.....	895
25.6.7	Buffer allocation.....	895
25.6.8	Reliability considerations.....	896
25.6.9	ECC on data flash accesses.....	898
25.6.10	Array integrity considerations.....	898

Chapter 26

Embedded Flash Memory (C55FMC)

26.1	Chip-specific C55FMC information.....	901
26.1.1	C55FMC register reset values.....	901
26.1.2	Access to and configuration of blocks mapped to C55FMC_LOCK1 register.....	901
26.1.3	C55FMC_LOCKn register bit mapping.....	902
26.1.4	C55FMC_SELn register bit mapping.....	905
26.1.5	C55FMC_OPpn register bit mapping.....	909
26.2	Introduction.....	912
26.2.1	Overview.....	913

Section number	Title	Page
26.2.2	Features.....	913
26.2.3	Modes of operation.....	913
26.3	UTest NVM block.....	914
26.4	Test mode disable seal.....	914
26.5	Memory map and register definition.....	916
26.5.1	Module Configuration Register (C55FMC_MCR).....	918
26.5.2	Extended Module Configuration Register (C55FMC_MCRE).....	923
26.5.3	Lock 0 register (C55FMC_LOCK0).....	926
26.5.4	Lock 1 register (C55FMC_LOCK1).....	928
26.5.5	Lock 2 register (C55FMC_LOCK2).....	928
26.5.6	Lock 3 register (C55FMC_LOCK3).....	929
26.5.7	Select 0 register (C55FMC_SEL0).....	930
26.5.8	Select 1 register (C55FMC_SEL1).....	931
26.5.9	Select 2 register (C55FMC_SEL2).....	932
26.5.10	Select 3 register (C55FMC_SEL3).....	933
26.5.11	Address register (C55FMC_ADR).....	934
26.5.12	UTest 0 register (C55FMC_UT0).....	936
26.5.13	UMISR register (C55FMC_UM n).....	939
26.5.14	UMISR register (C55FMC_UM9).....	940
26.5.15	Over-Program Protection 0 register (C55FMC_OPP0).....	941
26.5.16	Over-Program Protection 1 register (C55FMC_OPP1).....	942
26.5.17	Over-Program Protection 2 register (C55FMC_OPP2).....	943
26.5.18	Over-Program Protection 3 register (C55FMC_OPP3).....	943
26.5.19	Test Mode Disable Password Check register (C55FMC_TMD).....	944
26.6	Functional Description.....	945
26.6.1	User Mode.....	946
26.6.2	UTest mode.....	954
26.7	Initialization information.....	958

Section number	Title	Page
Chapter 27		
External Bus Interface (EBI)		
27.1	Chip-specific EBI information.....	961
27.1.1	EBI use cases.....	961
27.1.2	EBI signal names.....	961
27.1.3	EBI supported configurations.....	962
27.1.4	EBI CLKOUT division for slower-speed modes.....	963
27.1.5	EBI register reset values.....	963
27.1.6	EBI_TESR in Module Disable mode.....	963
27.2	Introduction.....	963
27.2.1	Overview.....	963
27.2.2	Features.....	964
27.2.3	Modes of Operation.....	964
27.3	External Signal Description.....	967
27.3.1	Overview.....	967
27.3.2	Detailed Signal Descriptions.....	968
27.3.3	Signal Output Buffer Enable Logic by Mode.....	971
27.4	Memory map and register definition.....	972
27.4.1	EBI Module Configuration Register (EBI_MCR).....	973
27.4.2	EBI Transfer Error Status Register (EBI_TESR).....	975
27.4.3	EBI Bus Monitor Control Register (EBI_BMCR).....	976
27.4.4	EBI Calibration Base Register (EBI_CAL_BR _n).....	977
27.4.5	EBI Calibration Option Register (EBI_CAL_OR _n).....	980
27.5	Functional Description.....	982
27.5.1	External Bus Interface Features.....	982
27.5.2	External Bus Operations.....	988
27.6	Initialization/Application Information.....	1037
27.6.1	Booting from External Memory.....	1037
27.6.2	Running with SDR (Single Data Rate) Burst Memories.....	1037

Section number	Title	Page
27.6.3	Running with Asynchronous Memories.....	1038
27.6.4	Connecting an MCU to Multiple Memories.....	1041
27.6.5	EBI Operation with Reduced Pinout MCUs.....	1042
27.6.6	Summary of Differences from MPC5xx.....	1043

Chapter 28 Analog Modules Overview

28.1	Introduction	1047
28.2	EQADC subsystem.....	1047
28.3	SDADC connections to other modules.....	1049

Chapter 29 Sigma Delta Analog-to-Digital Converter (SDADC) Digital Interface

29.1	Chip-specific SDADC information.....	1051
29.1.1	SDADC triggers and synchronous start.....	1051
29.1.2	Connections among SDADCs and eTPUs.....	1053
29.2	Introduction.....	1053
29.3	Overview.....	1053
29.4	Features.....	1054
29.5	Modes of operation.....	1055
29.5.1	Differential input mode.....	1055
29.5.2	Single-ended input mode.....	1055
29.6	External signal description.....	1056
29.7	Memory map and register definition.....	1056
29.7.1	Module Configuration Register (SDADC_MCR).....	1057
29.7.2	Channel Selection Register (SDADC_CSR).....	1060
29.7.3	Reset Key Register (SDADC_RKR).....	1062
29.7.4	Status Flag Register (SDADC_SFR).....	1063
29.7.5	Request Select and Enable Register (SDADC_RSER).....	1065
29.7.6	Output Settling Delay Register (SDADC_OSDR).....	1067
29.7.7	FIFO Control Register (SDADC_FCR).....	1068

Section number	Title	Page
29.7.8	Software Trigger Key Register (SDADC_STKR).....	1070
29.7.9	Converted Data Register (SDADC_CDR).....	1070
29.7.10	WDG Threshold Register (SDADC_WTHHLR).....	1071
29.8	Functional description.....	1071
29.8.1	Differential input mode.....	1072
29.8.2	Single-ended input mode.....	1072
29.8.3	Low consumption mode.....	1072
29.8.4	Analog input multiplexing and bias support.....	1072
29.8.5	Programmable gain and decimation rate.....	1073
29.8.6	High-pass filter support.....	1073
29.8.7	Data conversion.....	1073
29.8.8	Hardware triggering.....	1075
29.8.9	Interrupt/DMA request support.....	1076
29.8.10	Gain calibration support.....	1076
29.8.11	Offset calibration support.....	1077
29.9	Initialization information.....	1078
29.9.1	Data conversion step.....	1079

Chapter 30 Enhanced Queued Analog-to-Digital Converter (EQADC)

30.1	Chip-specific EQADC information.....	1081
30.1.1	EQADC connections to other modules.....	1082
30.1.2	EQADC Stop mode entry.....	1085
30.1.3	EQADC pre-gain control.....	1085
30.1.4	CFIFO and RFIFO depth.....	1085
30.1.5	EQADC pin pullup/pulldown control.....	1085
30.1.6	EQADC pin-to-channel mapping.....	1085
30.2	Introduction.....	1087
30.2.1	Module overview.....	1087
30.2.2	Block diagram.....	1088

Section number	Title	Page
30.2.3	Features.....	1090
30.3	Modes of operation.....	1091
30.3.1	Normal Mode.....	1091
30.3.2	Debug Mode.....	1092
30.3.3	Stop Mode.....	1093
30.4	External signal description.....	1094
30.4.1	Overview.....	1094
30.4.2	Detailed signal descriptions.....	1096
30.5	Memory map and register definition.....	1101
30.5.1	EQADC Module Configuration Register (EQADC_MCR).....	1106
30.5.2	EQADC Test Register (EQADC_TST).....	1107
30.5.3	EQADC External Trigger Digital Filter Register (EQADC_ETDFR).....	1108
30.5.4	EQADC CFIFO Push Register (EQADC_CFPRn).....	1109
30.5.5	EQADC Result FIFO Pop Register (EQADC_RFPRn).....	1110
30.5.6	EQADC CFIFO Control Register 0 (EQADC_CFCR0).....	1111
30.5.7	EQADC CFIFO Control Register 1 (EQADC_CFCR1).....	1115
30.5.8	EQADC CFIFO Control Register 2 (EQADC_CFCR2).....	1118
30.5.9	EQADC Interrupt and DMA Control Register 0 (EQADC_IDCR0).....	1121
30.5.10	EQADC Interrupt and DMA Control Register 1 (EQADC_IDCR1).....	1125
30.5.11	EQADC Interrupt and DMA Control Register 2 (EQADC_IDCR2).....	1130
30.5.12	EQADC FIFO and Interrupt Status Register (EQADC_FISRn).....	1135
30.5.13	EQADC CFIFO Transfer Counter Register 0 (EQADC_CFTCR0).....	1139
30.5.14	EQADC CFIFO Transfer Counter Register 1 (EQADC_CFTCR1).....	1140
30.5.15	EQADC CFIFO Transfer Counter Register 2 (EQADC_CFTCR2).....	1141
30.5.16	EQADC CFIFO Status Snapshot Register 0 (EQADC_CFSSR0).....	1142
30.5.17	EQADC CFIFO Status Snapshot Register 1 (EQADC_CFSSR1).....	1144
30.5.18	EQADC CFIFO Status Register (EQADC_CFSR).....	1146
30.5.19	EQADC STAC Bus Client Configuration Register (EQADC_STACCCR).....	1147
30.5.20	EQADC CFIFO0 Register (EQADC_CF0Rn).....	1148

Section number	Title	Page
30.5.21	EQADC CFIFO0 Extension Register (EQADC_CF0ERn).....	1149
30.5.22	EQADC CFIFO1 Register (EQADC_CF1Rn).....	1149
30.5.23	EQADC CFIFO2 Register (EQADC_CF2Rn).....	1150
30.5.24	EQADC CFIFO3 Register (EQADC_CF3Rn).....	1151
30.5.25	EQADC CFIFO4 Register (EQADC_CF4Rn).....	1151
30.5.26	EQADC CFIFO5 Register (EQADC_CF5Rn).....	1152
30.5.27	EQADC RFIFO0 Register (EQADC_RF0Rn).....	1153
30.5.28	EQADC RFIFO1 Register (EQADC_RF1Rn).....	1153
30.5.29	EQADC RFIFO2 Register (EQADC_RF2Rn).....	1154
30.5.30	EQADC RFIFO3 Register (EQADC_RF3Rn).....	1155
30.5.31	EQADC RFIFO4 Register (EQADC_RF4Rn).....	1155
30.5.32	EQADC RFIFO5 Register (EQADC_RF5Rn).....	1156
30.5.33	On-chip ADC registers.....	1156
30.6	Functional description.....	1181
30.6.1	Overview.....	1181
30.6.2	Data flow in EQADC.....	1182
30.6.3	Command/result queues.....	1195
30.6.4	EQADC command FIFOs.....	1195
30.6.5	EQADC result FIFOs.....	1223
30.6.6	On-chip ADC configuration and control.....	1227
30.6.7	Internal/external multiplexing.....	1240
30.6.8	EQADC DMA/interrupt request.....	1244
30.6.9	EQADC Parallel Side Interface (PSI) sub-block.....	1247
30.6.10	Analog sub-block.....	1252
30.7	Initialization/application information.....	1255
30.7.1	Multiple queues control setup example.....	1255
30.7.2	EQADC/DMAC interface.....	1260
30.7.3	Sending immediate command setup example.....	1262
30.7.4	Modifying queues.....	1263

Section number	Title	Page
30.7.5	CQueue and RQueues usage.....	1264
30.7.6	ADC Result Calibration.....	1266
30.7.7	EQADC versus QADC.....	1269

Chapter 31 Decimation Filter (DECFILTER)

31.1	Chip-specific DECFILTER information.....	1273
31.1.1	DECFILTER instance interconnections.....	1273
31.1.2	DECFILTER connections to other modules.....	1274
31.2	Introduction.....	1275
31.2.1	Overview.....	1276
31.2.2	Features.....	1279
31.2.3	Modes of operation.....	1279
31.3	External signal description.....	1282
31.3.1	Decimation trigger signal	1282
31.3.2	Integrator enable signal	1282
31.3.3	Integrator halt signal	1282
31.3.4	Integrator reset signal	1282
31.3.5	Integrator output request signal	1282
31.4	Memory map and register definition.....	1283
31.4.1	Module Configuration Register (DECFILTER_MCR).....	1284
31.4.2	Module Status Register (DECFILTER_MSR).....	1290
31.4.3	Module Extended Config Register (DECFILTER_MXCR).....	1294
31.4.4	Module Extended Status Register (DECFILTER_MXSR).....	1297
31.4.5	Interface Input Buffer (DECFILTER_IB).....	1300
31.4.6	Interface Output Buffer (DECFILTER_OB).....	1302
31.4.7	Filter Coefficient (DECFILTER_COEF _n).....	1303
31.4.8	Filter TAP (DECFILTER_TAP _n).....	1303
31.4.9	Enhanced Debug Input Data (DECFILTER_EDID).....	1304
31.4.10	Final Integer Value Register (DECFILTER_FINTVAL).....	1305

Section number	Title	Page
31.4.11	Final Integer Count Register (DECFILTER_FINTCNT).....	1305
31.4.12	Current Integer Value Register (DECFILTER_CINTVAL).....	1306
31.4.13	Current Integer Count Register (DECFILTER_CINTCNT).....	1306
31.5	Functional Description.....	1307
31.5.1	Overview.....	1307
31.5.2	Parallel Side Interface (PSI) sub-block description.....	1307
31.5.3	Input Buffer Description.....	1308
31.5.4	Output Buffer Description.....	1309
31.5.5	Bypass Configuration Description.....	1312
31.5.6	IIR and FIR Filter.....	1313
31.5.7	Filter Prefill Control Description.....	1316
31.5.8	Timestamp Data Transmission.....	1317
31.5.9	Flush Command Description.....	1318
31.5.10	Soft-Reset Command Description.....	1319
31.5.11	Interrupts Requests Description.....	1320
31.5.12	DMA Requests Description.....	1322
31.5.13	Freeze Mode Description.....	1323
31.5.14	Enhanced Debug Monitor Description.....	1324
31.5.15	Integrator.....	1324
31.5.16	Cascade mode description.....	1329
31.5.17	Return PSI (RPSI) and PSI Output Selector (PSIOSEL).....	1334
31.6	Initialization Information.....	1334
31.6.1	Initialization Procedure.....	1334
31.7	Application Information.....	1335
31.7.1	eQADC IP as the PSI Master Block.....	1335
31.8	Filter Example Simulation.....	1336
31.8.1	Coefficients Calculation.....	1336
31.8.2	Input Data Calculation.....	1338
31.8.3	Filter Results.....	1338

Section number	Title	Page
Chapter 32		
Temperature Sensor (TSENS)		
32.1	Chip-specific Temperature Sensor information.....	1339
32.1.1	Temperature Sensor programming control.....	1339
32.1.2	Accessing TSC parameters.....	1339
32.1.3	Temperature name alignment.....	1339
32.2	Introduction.....	1340
32.3	Functional Description.....	1340
32.3.1	Temperature threshold detection (digital output generation).....	1340
32.3.2	Linear temperature sensor (analog output generation).....	1341
32.4	Temperature formula.....	1342
32.5	Calculating device temperature.....	1343
Chapter 33		
System Timer Module (STM)		
33.1	Chip-specific STM information.....	1345
33.2	Introduction.....	1345
33.2.1	Overview.....	1345
33.2.2	Features.....	1345
33.2.3	Modes of operation.....	1346
33.3	External signal description.....	1346
33.4	Memory map and registers.....	1346
33.4.1	STM Control Register (STM_CR).....	1347
33.4.2	STM Count Register (STM_CNT).....	1348
33.4.3	STM Channel Control Register (STM_CCR _n).....	1348
33.4.4	STM Channel Interrupt Register (STM_CIR _n).....	1349
33.4.5	STM Channel Compare Register (STM_CMP _n).....	1349
33.5	Functional description.....	1350

Chapter 34

Software Watchdog Timer (SWT)

34.1	Chip-specific SWT information.....	1351
34.1.1	SWT register reset values.....	1351
34.1.2	SWT time-out events and invalid accesses: configuring reset and interrupt requests.....	1351
34.2	Introduction.....	1352
34.2.1	Overview.....	1352
34.2.2	Features.....	1353
34.2.3	Modes of operation.....	1353
34.3	External signal description.....	1354
34.4	Memory Map and Registers.....	1354
34.4.1	SWT Control Register (SWT_CR).....	1355
34.4.2	SWT Interrupt Register (SWT_IR).....	1358
34.4.3	SWT Time-out Register (SWT_TO).....	1359
34.4.4	SWT Window Register (SWT_WN).....	1359
34.4.5	SWT Service Register (SWT_SR).....	1360
34.4.6	SWT Counter Output Register (SWT_CO).....	1360
34.4.7	SWT Service Key Register (SWT_SK).....	1361
34.5	Functional description.....	1361
34.5.1	Introduction.....	1361
34.5.2	Configuration locking.....	1362
34.5.3	Unlock sequence.....	1363
34.5.4	Servicing operations.....	1363
34.5.5	Time-out.....	1364
34.5.6	Initialization.....	1365

Chapter 35

Periodic Interrupt Timer (PIT-RTI)

35.1	Chip-specific PIT information.....	1367

Section number	Title	Page
35.2	Introduction.....	1367
35.2.1	Block diagram.....	1367
35.2.2	Features.....	1368
35.3	Signal description.....	1369
35.4	Memory map/register description.....	1369
35.4.1	PIT Module Control Register (PIT_MCR).....	1370
35.4.2	PIT Upper Lifetime Timer Register (PIT_LTMR64H).....	1371
35.4.3	PIT Lower Lifetime Timer Register (PIT_LTMR64L).....	1372
35.4.4	Timer Load Value Register (PIT_RTI_LDVAL).....	1372
35.4.5	Current Timer Value Register (PIT_RTI_CVAL).....	1373
35.4.6	Timer Control Register (PIT_RTI_TCTRL).....	1373
35.4.7	Timer Flag Register (PIT_RTI_TFLG).....	1374
35.4.8	Timer Load Value Register (PIT_LDVAL _n).....	1375
35.4.9	Current Timer Value Register (PIT_CVAL _n).....	1375
35.4.10	Timer Control Register (PIT_TCTRL _n).....	1376
35.4.11	Timer Flag Register (PIT_TFLG _n).....	1377
35.5	Functional description.....	1377
35.5.1	General operation.....	1377
35.5.2	Interrupts.....	1379
35.5.3	Chained timers.....	1379
35.6	Initialization and application information.....	1380
35.7	Example configuration for chained timers.....	1381
35.8	Example configuration for the lifetime timer.....	1382

Chapter 36

Enhanced Modular IO Subsystem (eMIOS)

36.1	Chip-specific eMIOS information.....	1383
36.1.1	eMIOS channels and counter buses.....	1383
36.1.2	eMIOS unimplemented channels.....	1383

Section number	Title	Page
36.1.3	eMIOS interaction with other modules.....	1384
36.2	Introduction.....	1386
36.2.1	Overview.....	1386
36.2.2	Features.....	1387
36.2.3	Modes of operation.....	1388
36.3	Memory map / register definition.....	1389
36.3.1	Memory map.....	1389
36.4	Control Memory Map and Registers.....	1390
36.4.1	eMIOS Module Configuration Register (eMIOS_MCR).....	1391
36.4.2	eMIOS Global FLAG Register (eMIOS_GFLAG).....	1393
36.4.3	eMIOS Output Update Disable register (eMIOS_OUDIS).....	1393
36.5	Unified Channels Memory Map and Registers.....	1394
36.5.1	eMIOS UC A register n (eMIOS_An).....	1397
36.5.2	eMIOS UC B register n (eMIOS_Bn).....	1398
36.5.3	eMIOS CNT register n (eMIOS_CNTn).....	1399
36.5.4	eMIOS UC Control register n (eMIOS_Cn).....	1400
36.5.5	eMIOS UC Status register n (eMIOS_Sn).....	1406
36.5.6	eMIOS Alternate Address register n (eMIOS_ALTAn).....	1408
36.5.7	eMIOS UC A register n (eMIOS_An).....	1409
36.5.8	eMIOS UC B register n (eMIOS_Bn).....	1409
36.5.9	eMIOS CNT register n (eMIOS_CNTn).....	1410
36.5.10	eMIOS UC Control register n (eMIOS_Cn).....	1411
36.5.11	eMIOS UC Status register n (eMIOS_Sn).....	1416
36.5.12	eMIOS Alternate Address register n (eMIOS_ALTAn).....	1418

Section number	Title	Page
36.6	Functional Description.....	1418
36.6.1	Unified Channel (UC).....	1420
36.6.2	IP Bus Interface Unit (BIU).....	1465
36.6.3	STAC Bus Client submodule.....	1466
36.6.4	Global Clock Prescaler Submodule (GCP).....	1467
36.7	Initialization/Application Information.....	1467
36.7.1	Considerations.....	1467
36.7.2	Application Information.....	1468

Chapter 37

Enhanced Time Processing Unit (eTPU)

37.1	Chip-specific eTPU information.....	1473
37.1.1	eTPU register field reset values.....	1474
37.1.2	Access to eTPU registers in chip-level Stop mode.....	1474
37.1.3	STAC server IDs.....	1474
37.1.4	eTPU inputs with Glitch Filters.....	1475
37.1.5	eMIOS and eTPU signal passthrough from pins via IGFs.....	1475
37.1.6	eTPU output disable channel groups.....	1476
37.1.7	Control of the eTPU Output Buffer Enable signal.....	1476
37.2	Device-specific Information.....	1476
37.3	Introduction.....	1477
37.3.1	Overview.....	1478
37.3.2	Features.....	1485
37.3.3	Modes of operation.....	1492
37.4	Detailed signal descriptions.....	1494
37.4.1	eTPU channel output signals.....	1494
37.4.2	eTPU channel input signals.....	1494
37.4.3	Time Base Clock Signal — TCRCLK.....	1495
37.4.4	eTPU channel output disable signals.....	1495
37.5	Memory map/register definition.....	1496

Section number	Title	Page
37.6	Memory map.....	1496
37.7	System Configuration memory map and registers.....	1501
37.7.1	eTPU Module Configuration Register (eTPU_SCR_MCR).....	1502
37.7.2	eTPU Coherent Dual-Parameter Controller Register (eTPU_SCR_CDCR).....	1506
37.7.3	eTPU MISC Compare Register (eTPU_SCR_MISCCMPR).....	1508
37.7.4	eTPU SCM Off-range Data Register (eTPU_SCR_SCMOFFDATAR).....	1508
37.7.5	eTPU Engine 1 Configuration Register (eTPU_SCR_ECR_ENG1).....	1509
37.7.6	eTPU Engine 2 Configuration Register (eTPU_SCR_ECR_ENG2).....	1513
37.8	Timebase memory map and registers.....	1517
37.8.1	eTPU Engine 1 Time Base Configuration Register (eTPU_TBR_TBCR_ENG1).....	1518
37.8.2	eTPU Engine 1 Time Base 1 (TCR1) Visibility Register (eTPU_TBR_TB1R_ENG1).....	1522
37.8.3	eTPU Engine 1 Time Base 2 (TCR2) Visibility Register (eTPU_TBR_TB2R_ENG1).....	1522
37.8.4	eTPU Engine 1 STAC Configuration Register (eTPU_TBR_STAC_ENG1).....	1523
37.8.5	eTPU Engine 2 Time Base Configuration Register (eTPU_TBR_TBCR_ENG2).....	1525
37.8.6	eTPU Engine 2 Time Base 1 (TCR1) Visibility Register (eTPU_TBR_TB1R_ENG2).....	1529
37.8.7	eTPU Engine 2 Time Base 2 (TCR2) Visibility Register (eTPU_TBR_TB2R_ENG2).....	1529
37.8.8	eTPU Engine 2 STAC Configuration Register (eTPU_TBR_STAC_ENG2).....	1530
37.9	Engine memory map and registers.....	1532
37.9.1	eTPU Watchdog Timer Register for Engine 1 (eTPU_ER_WDTR_ENG1).....	1532
37.9.2	eTPU Idle Register for Engine 1 (eTPU_ER_IDLER_ENG1).....	1534
37.9.3	eTPU Watchdog Timer Register for Engine 2 (eTPU_ER_WDTR_ENG2).....	1535
37.9.4	eTPU Idle Register for Engine 2 (eTPU_ER_IDLER_ENG2).....	1536
37.10	Latency Monitor memory map and registers.....	1537
37.10.1	eTPU Latency Monitor Configuration Register for Engine 1 (eTPU_LMR_LMnCR_ENG1).....	1538
37.10.2	eTPU Latency Monitor Measure Register for Engine 1 (eTPU_LMR_LMnMR_ENG1).....	1540
37.10.3	eTPU Latency Monitor Configuration Register for Engine 2 (eTPU_LMR_LMnCR_ENG2).....	1541
37.10.4	eTPU Latency Monitor Measure Register for Engine 2 (eTPU_LMR_LMnMR_ENG2).....	1542
37.11	Memory Error Support memory map and registers.....	1543
37.11.1	eTPU Memory Error Control Register (eTPU_MESR_MECR).....	1544

Section number	Title	Page
37.11.2	eTPU Data Error Injection Address Register (eTPU_MESR_DEIAR).....	1547
37.11.3	eTPU Data Error Injection Data Pattern Register (eTPU_MESR_DEIDPR).....	1548
37.11.4	eTPU Data Error Injection Parity Pattern Register (eTPU_MESR_DEIPPR).....	1548
37.11.5	eTPU Data Error Report Address Register (eTPU_MESR_DERAR).....	1550
37.11.6	eTPU Data Error Report Data Register (eTPU_MESR_DERDR).....	1551
37.11.7	eTPU Data Error Report Syndrome Register (eTPU_MESR_DERSR).....	1552
37.11.8	eTPU Memory Error Status Register (eTPU_MESR_MESR).....	1554
37.11.9	eTPU Code Error Injection Address Register (eTPU_MESR_CEIAR).....	1557
37.11.10	eTPU Code Error Injection Data Pattern Register (eTPU_MESR_CEIDPR).....	1558
37.11.11	eTPU Code Error Injection Parity Pattern Register (eTPU_MESR_CEIPPR).....	1558
37.11.12	eTPU Code Error Report Address Register (eTPU_MESR_CERAR).....	1559
37.11.13	eTPU Code Error Report Data Register (eTPU_MESR_CERDR).....	1560
37.11.14	eTPU Code Error Report Syndrome Register (eTPU_MESR_CERSR).....	1561
37.11.15	eTPU Code Error Fix Register (eTPU_MESR_CEFR).....	1562
37.12	Global Channel memory map and registers.....	1563
37.12.1	eTPU Channel Interrupt Status Register for Engine 1 (eTPU_GCR_CISR_ENG1).....	1564
37.12.2	eTPU Channel Interrupt Status Register for Engine 2 (eTPU_GCR_CISR_ENG2).....	1564
37.12.3	eTPU Channel Data Transfer Request Status Register for Engine 1 (eTPU_GCR_CDTRSR_ENG1).....	1565
37.12.4	eTPU Channel Data Transfer Request Status Register for Engine 2 (eTPU_GCR_CDTRSR_ENG2).....	1566
37.12.5	eTPU Channel Interrupt Overflow Status Register for Engine 1 (eTPU_GCR_CIOSR_ENG1).....	1566
37.12.6	eTPU Channel Interrupt Overflow Status Register for Engine 2 (eTPU_GCR_CIOSR_ENG2).....	1567
37.12.7	eTPU Channel Data Transfer Request Overflow Status Register for Engine 1 (eTPU_GCR_CDTROSR_ENG1).....	1568
37.12.8	eTPU Channel Data Transfer Request Overflow Status Register for Engine 2 (eTPU_GCR_CDTROSR_ENG2).....	1568
37.12.9	eTPU Channel Interrupt Enable Register for Engine 1 (eTPU_GCR_CIER_ENG1).....	1569
37.12.10	eTPU Channel Interrupt Enable Register for Engine 2 (eTPU_GCR_CIER_ENG2).....	1569
37.12.11	eTPU Channel Data Transfer Request Enable Register for Engine 1 (eTPU_GCR_CDTRER_ENG1)....	1570
37.12.12	eTPU Channel Data Transfer Request Enable Register for Engine 2 (eTPU_GCR_CDTRER_ENG2)....	1570

Section number	Title	Page
37.12.13	eTPU Watchdog Status Register for Engine 1 (eTPU_GCR_WDSR_ENG1).....	1571
37.12.14	eTPU Watchdog Status Register for Engine 2 (eTPU_GCR_WDSR_ENG2).....	1572
37.12.15	eTPU Channel Pending Service Status Register for Engine 1 (eTPU_GCR_CPSSR_ENG1).....	1572
37.12.16	eTPU Channel Pending Service Status Register for Engine 2 (eTPU_GCR_CPSSR_ENG2).....	1573
37.12.17	eTPU Channel Service Status Register for Engine 1 (eTPU_GCR_CSSR_ENG1).....	1574
37.12.18	eTPU Channel Service Status Register for Engine 2 (eTPU_GCR_CSSR_ENG2).....	1574
37.13	Channel Configuration Control memory map and registers.....	1575
37.13.1	eTPU Channel n Configuration Register for Engine 1 (eTPU_CCCR_CnCR_ENG1).....	1585
37.13.2	eTPU Channel n Status Control Register for Engine 1 (eTPU_CCCR_CnSCR_ENG1).....	1588
37.13.3	eTPU Channel n Host Service Request Register for Engine 1 (eTPU_CCCR_CnHSRR_ENG1).....	1590
37.13.4	eTPU Channel n Configuration Register for Engine 2 (eTPU_CCCR_CnCR_ENG2).....	1591
37.13.5	eTPU Channel n Status Control Register for Engine 2 (eTPU_CCCR_CnSCR_ENG2).....	1594
37.13.6	eTPU Channel n Host Service Request Register for Engine 2 (eTPU_CCCR_CnHSRR_ENG2).....	1596
37.14	Shared Subsystem Integration memory map and registers.....	1597
37.14.1	eTPU Shared Subsystem n Integration Register (eTPU_SSIR_ETPUSSSIRn).....	1598
37.15	Functional description.....	1598
37.15.1	Functions and threads.....	1599
37.15.2	Host interface.....	1614
37.15.3	Scheduler.....	1623
37.15.4	Parameter sharing and coherency.....	1631
37.15.5	Enhanced Channels.....	1637
37.15.6	Time bases.....	1691
37.15.7	EAC - eTPU Angle Counter.....	1700
37.15.8	Microengine.....	1723
37.15.9	Microinstruction Set.....	1742
37.15.10	Shared Subsystems support.....	1776
37.15.11	Development support features.....	1780
37.15.12	SCM Emulation.....	1788

Section number	Title	Page
37.15.13	Safety features.....	1788
37.15.14	Performance monitoring features.....	1796
37.16	Initialization/application information.....	1800
37.16.1	Configuration sequence.....	1800
37.16.2	Reset options.....	1802
37.16.3	Multiple parameter coherency methods.....	1802
37.16.4	Programming Hints and Caveats.....	1803
37.16.5	Estimating Worst Case Latency.....	1805
37.16.6	Endianness.....	1823
37.16.7	Memory Error Service Procedures.....	1824
37.17	Appendices.....	1827
37.17.1	Microcycle and I/O timing.....	1827
37.17.2	Channel Mode Summary.....	1832
37.17.3	MISC Algorithm.....	1835

Chapter 38

Reaction Module 2 (REACM2)

38.1	Chip-specific REACM2 information.....	1837
38.1.1	REACM2 interaction with other modules.....	1837
38.1.2	Timer inputs.....	1838
38.1.3	REACM2 buffer overflow prevention.....	1838
38.2	Introduction.....	1839
38.2.1	Features.....	1839
38.2.2	Modes of Operation.....	1839
38.2.3	Block Diagram.....	1842
38.3	Signal Description.....	1846
38.3.1	REACM2_RCHn — REACM Channel (n) Output Pin a, b and c.....	1846
38.4	Memory Map and Register Definition.....	1847
38.4.1	Module Configuration (REACM2_MCR).....	1852
38.4.2	Timer Configuration (REACM2_TCR).....	1854

Section number	Title	Page
38.4.3	Threshold Router Register (REACM2_THRR).....	1854
38.4.4	ADC Sensor Input Register (REACM2_SINR).....	1856
38.4.5	Period Generator Configuration Register (REACM2_PCR).....	1857
38.4.6	Period Shift Delay Configuration Register (REACM2_PSCR).....	1857
38.4.7	Global Error Flag Register (REACM2_GEFR).....	1858
38.4.8	Channel n Configuration Register (REACM2_CHCR _n).....	1860
38.4.9	Channel n Status Register (REACM2_CHSR _n).....	1863
38.4.10	Channel n Router Register (REACM2_CHRR _n).....	1866
38.4.11	Shared Timer Bank (REACM2_STBK _n).....	1867
38.4.12	Hold Timer Bank (REACM2_HOTBK _n).....	1867
38.4.13	Threshold Bank (REACM2_THBK _n).....	1868
38.4.14	ADC Result Maximum Limit Check (REACM2_ADCMAX).....	1869
38.4.15	Modulation Range Pulse Width (REACM2_RANGE_PWD).....	1869
38.4.16	Modulation Minimum Pulse Width (REACM2_MIN_PWD).....	1870
38.4.17	Modulation Control Word Bank (REACM2_MWBK _n).....	1871
38.5	Functional Description.....	1873
38.5.1	Reaction Channel.....	1874
38.5.2	Modulation Control Words Bank.....	1876
38.5.3	Shared Timer Bank.....	1877
38.5.4	Hold Time Bank.....	1879
38.5.5	Threshold Bank and Comparator.....	1879
38.5.6	ADC Interface.....	1880
38.5.7	Prescalers.....	1883
38.5.8	Period / Delay Generator.....	1883
38.5.9	Banked Mode Support.....	1884
38.6	Modulation Modes.....	1885
38.6.1	Threshold/Threshold Mode.....	1885
38.6.2	Threshold/Hold-off Mode.....	1886
38.6.3	Threshold/Hold-on Mode.....	1887

Section number	Title	Page
38.6.4	Threshold low/Period Mode.....	1888
38.6.5	Threshold high/Period Mode.....	1889
38.6.6	Limitations on the Modulation Process.....	1890
38.7	Monitored Modulation.....	1894
38.7.1	Monitoring the ON Pulse Width.....	1894
38.7.2	Monitoring the OFF State Width and the ADC Maximum Value.....	1898
38.8	Reset Overview.....	1900
38.9	Reaction Module Interrupts.....	1901
38.9.1	Interrupt sources.....	1901
38.9.2	Global Interrupt and Channel Interrupt.....	1901
38.10	Use cases.....	1902
38.10.1	Advancing Modulation Phase on a Threshold level.....	1909
38.10.2	Controlling the LOOP function.....	1910
38.10.3	Banked Mode.....	1911

Chapter 39 Input Glitch Filter (IGF)

39.1	Chip-specific IGF information.....	1913
39.1.1	IGF channels and internal prescalers.....	1913
39.1.2	IGF internal prescaler selection.....	1913
39.1.3	IGF interaction with other modules.....	1914
39.1.4	STOP mode system requests for IGF_0 and IGF_1.....	1915
39.2	Introduction.....	1915
39.3	Block Diagram.....	1916
39.4	Input Glitch Filter overview.....	1916
39.4.1	Interaction with other peripherals.....	1917
39.4.2	Mode of operation.....	1917
39.5	Memory map.....	1919
39.5.1	Module Configuration Register (IGF_MCR _n).....	1924
39.5.2	Module Status Register (IGF_MSR _n).....	1927

Section number	Title	Page
39.5.3	Prescaler Register (IGF_PRESR _n).....	1928
39.5.4	Rising edge Threshold Register (IGF_RTHR _n).....	1929
39.5.5	Falling edge Threshold Register (IGF_FTHR).....	1929
39.6	Functional description.....	1929
39.6.1	Input double sampling.....	1930
39.6.2	Filter types.....	1930
39.6.3	Application.....	1934
39.6.4	Programming guidelines.....	1937
39.6.5	On-the-fly filter reconfiguration.....	1937
39.7	Power saving mode.....	1938
39.7.1	STOP mode.....	1938

Chapter 40 FlexCAN (CAN)

40.1	Chip-specific FlexCAN information.....	1941
40.1.1	FlexCAN RAM.....	1941
40.1.2	FlexCAN reset.....	1941
40.1.3	CAN_MCR[SUPV] and access to FlexCAN registers.....	1941
40.2	Introduction.....	1941
40.2.1	Overview.....	1942
40.2.2	FlexCAN module features.....	1943
40.2.3	Modes of operation.....	1945
40.3	FlexCAN signal descriptions.....	1946
40.3.1	CAN Rx	1946
40.3.2	CAN Tx	1947
40.4	Memory map/register definition.....	1947
40.4.1	FlexCAN memory mapping.....	1947
40.4.2	Module Configuration Register (CAN_MCR).....	1953
40.4.3	Control 1 register (CAN_CTRL1).....	1958
40.4.4	Free Running Timer (CAN_TIMER).....	1961

Section number	Title	Page
40.4.5	Rx Mailboxes Global Mask Register (CAN_RXMGMASK).....	1962
40.4.6	Rx 14 Mask register (CAN_RX14MASK).....	1963
40.4.7	Rx 15 Mask register (CAN_RX15MASK).....	1964
40.4.8	Error Counter (CAN_ECR).....	1964
40.4.9	Error and Status 1 register (CAN_ESR1).....	1966
40.4.10	Interrupt Masks 2 register (CAN_IMASK2).....	1970
40.4.11	Interrupt Masks 1 register (CAN_IMASK1).....	1970
40.4.12	Interrupt Flags 2 register (CAN_IFLAG2).....	1971
40.4.13	Interrupt Flags 1 register (CAN_IFLAG1).....	1972
40.4.14	Control 2 register (CAN_CTRL2).....	1974
40.4.15	Error and Status 2 register (CAN_ESR2).....	1978
40.4.16	CRC Register (CAN_CRCR).....	1979
40.4.17	Rx FIFO Global Mask register (CAN_RXFGMASK).....	1980
40.4.18	Rx FIFO Information Register (CAN_RXFIR).....	1981
40.4.19	Rx Individual Mask Registers (CAN_RXIMR _n).....	1981
40.4.20	Memory Error Control Register (CAN_MECR).....	1982
40.4.21	Error Injection Address Register (CAN_ERRIAR).....	1984
40.4.22	Error Injection Data Pattern Register (CAN_ERRIDPR).....	1986
40.4.23	Error Injection Parity Pattern Register (CAN_ERRIPPR).....	1986
40.4.24	Error Report Address Register (CAN_RERRAR).....	1987
40.4.25	Error Report Data Register (CAN_RERRDR).....	1988
40.4.26	Error Report Syndrome Register (CAN_RERRSYNR).....	1989
40.4.27	Error Status Register (CAN_ERRSR).....	1991
40.4.28	Message buffer structure.....	1992
40.4.29	Rx FIFO structure.....	1998
40.5	Functional description.....	2000
40.5.1	Transmit process.....	2001
40.5.2	Arbitration process.....	2002
40.5.3	Receive process.....	2005

Section number	Title	Page
40.5.4	Matching process.....	2007
40.5.5	Move process.....	2012
40.5.6	Data coherence.....	2014
40.5.7	Rx FIFO.....	2017
40.5.8	CAN protocol related features.....	2019
40.5.9	Clock domains and restrictions.....	2025
40.5.10	Modes of operation details.....	2026
40.5.11	Interrupts.....	2030
40.5.12	Bus interface.....	2031
40.5.13	Detection and Correction of Memory Errors.....	2031
40.6	Initialization/application information.....	2035
40.6.1	FlexCAN initialization sequence.....	2035

Chapter 41 Modular CAN (M_CAN)

41.1	Chip-specific M_CAN information.....	2039
41.1.1	M_CAN Message RAM allocation.....	2039
41.1.2	Introduction.....	2039
41.1.3	Functional Description.....	2040
41.1.4	External Signals.....	2042
41.2	Overview.....	2043
41.2.1	Features.....	2043
41.2.2	Block Diagram.....	2044
41.2.3	Dual Clock Sources.....	2046
41.2.4	Dual Interrupt Lines.....	2046
41.3	Memory Map and Register Description.....	2047
41.3.1	Core Release Register (M_CAN_CREL).....	2049
41.3.2	Endian Register (M_CAN_ENDN).....	2050
41.3.3	Data Bit Timing and Prescaler Register (M_CAN_DBTP).....	2051
41.3.4	Test Register (M_CAN_TEST).....	2052

Section number	Title	Page
41.3.5	RAM Watchdog Register (M_CAN_RWD).....	2054
41.3.6	CC Control Register (M_CAN_CCCR).....	2055
41.3.7	Nominal Bit Timing and Prescaler Register (M_CAN_NBTP).....	2057
41.3.8	Timestamp Counter Configuration Register (M_CAN_TSCC).....	2059
41.3.9	Timestamp Counter Value Register (M_CAN_TSCV).....	2060
41.3.10	Timeout Counter Configuration Register (M_CAN_TOCC).....	2060
41.3.11	Timeout Counter Value Register (M_CAN_TOCV).....	2061
41.3.12	Error Counter Register (M_CAN_ECR).....	2062
41.3.13	Protocol Status Register (M_CAN_PSR).....	2063
41.3.14	Transmitter Delay Compensation Register (M_CAN_TDCR).....	2066
41.3.15	Interrupt Register (M_CAN_IR).....	2067
41.3.16	Interrupt Enable Register (M_CAN_IE).....	2070
41.3.17	Interrupt Line Select Register (M_CAN_ILS).....	2073
41.3.18	Interrupt Line Enable Register (M_CAN_ILE).....	2077
41.3.19	Global Filter Configuration Register (M_CAN_GFC).....	2078
41.3.20	Standard ID Filter Configuration Register (M_CAN_SIDFC).....	2079
41.3.21	Extended ID Filter Configuration Register (M_CAN_XIDFC).....	2080
41.3.22	Extended ID and Mask Register (M_CAN_XIDAM).....	2080
41.3.23	High Priority Message Status Register (M_CAN_HPMS).....	2081
41.3.24	New Data 1 Register (M_CAN_NDAT1).....	2082
41.3.25	New Data 2 Register (M_CAN_NDAT2).....	2082
41.3.26	Rx FIFO 0 Configuration Register (M_CAN_RXF0C).....	2083
41.3.27	Rx FIFO 0 Status Register (M_CAN_RXF0S).....	2084
41.3.28	Rx FIFO 0 Acknowledge Register (M_CAN_RXF0A).....	2085
41.3.29	Rx Buffer Configuration Register (M_CAN_RXBC).....	2085
41.3.30	Rx FIFO 1 Configuration Register (M_CAN_RXF1C).....	2086
41.3.31	Rx FIFO 1 Status Register (M_CAN_RXF1S).....	2087
41.3.32	Rx FIFO 1 Acknowledge Register (M_CAN_RXF1A).....	2088
41.3.33	Rx Buffer / FIFO Element Size Configuration Register (M_CAN_RXESC).....	2089

Section number	Title	Page
41.3.34	Tx Buffer Configuration Register (M_CAN_TXBC).....	2091
41.3.35	Tx FIFO/Queue Status Register (M_CAN_TXFQS).....	2092
41.3.36	Tx Buffer Element Size Configuration (M_CAN_TXESC).....	2093
41.3.37	Tx Buffer Request Pending Register (M_CAN_TXBRP).....	2094
41.3.38	Tx Buffer Add Request Register (M_CAN_TXBAR).....	2095
41.3.39	Tx Buffer Cancellation Request Register (M_CAN_TXBCR).....	2095
41.3.40	Tx Buffer Transmission Occurred Register (M_CAN_TXBTO).....	2096
41.3.41	Tx Buffer Cancellation Finished Register (M_CAN_TXBCF).....	2096
41.3.42	Tx Buffer Transmission Interrupt Enable Register (M_CAN_TXBTIE).....	2097
41.3.43	Tx Buffer Cancellation Finished Interrupt Enable Register (M_CAN_TXBCIE).....	2097
41.3.44	Tx Event FIFO Configuration Register (M_CAN_TXEFC).....	2098
41.3.45	Tx Event FIFO Status Register (M_CAN_TXEFS).....	2099
41.3.46	Tx Event FIFO Acknowledge Register (M_CAN_TXEFA).....	2100
41.4	Message RAM.....	2100
41.4.1	Rx Buffer and FIFO Element.....	2101
41.4.2	Tx Buffer Element.....	2103
41.4.3	Tx Event FIFO Element.....	2105
41.4.4	Standard Message ID Filter Element.....	2106
41.4.5	Extended Message ID Filter Element.....	2108
41.5	Functional Description.....	2110
41.5.1	Operating Modes.....	2110
41.5.2	Timestamp Generation.....	2119
41.5.3	Timeout Counter.....	2120
41.5.4	Rx Handling.....	2120
41.5.5	Tx Handling.....	2131
41.5.6	FIFO Acknowledge Handling.....	2137

Section number	Title	Page
Chapter 42		
Deserial Serial Peripheral Interface (DSPI)		
42.1	Chip-specific DSPI information.....	2139
42.1.1	DSPI and eMIOS direct connections.....	2139
42.1.2	DSPI connections with SIU, eTPU, and eMIOS modules.....	2140
42.2	Introduction.....	2141
42.2.1	Block Diagram.....	2141
42.2.2	Features.....	2142
42.2.3	Interface configurations.....	2145
42.2.4	Modes of Operation.....	2146
42.3	Module signal descriptions.....	2147
42.3.1	HT—Hardware Trigger.....	2148
42.3.2	PCS0/SS—Peripheral Chip Select/Slave Select.....	2148
42.3.3	PCS1–PCS3—Peripheral Chip Selects 1–3.....	2148
42.3.4	PCS4—Peripheral Chip Select 4.....	2149
42.3.5	PCS5/PCSS—Peripheral Chip Select 5/Peripheral Chip Select Strobe.....	2149
42.3.6	SCK—Serial Clock.....	2149
42.3.7	SIN—Serial Input.....	2149
42.3.8	SOUT—Serial Output.....	2149
42.4	Memory Map/Register Definition.....	2150
42.4.1	Module Configuration Register (DSPI_MCR).....	2152
42.4.2	Transfer Count Register (DSPI_TCR).....	2155
42.4.3	Clock and Transfer Attributes Register (In Master Mode) (DSPI_CTAR _n).....	2156
42.4.4	Clock and Transfer Attributes Register (In Slave Mode) (DSPI_CTAR _n _SLAVE).....	2161
42.4.5	Status Register (DSPI_SR).....	2163
42.4.6	DMA/Interrupt Request Select and Enable Register (DSPI_RSER).....	2166
42.4.7	PUSH TX FIFO Register In Master Mode (DSPI_PUSHHR).....	2168
42.4.8	PUSH TX FIFO Register In Slave Mode (DSPI_PUSHHR_SLAVE).....	2170
42.4.9	POP RX FIFO Register (DSPI_POPR).....	2171

Section number	Title	Page
42.4.10	Transmit FIFO Registers (DSPI_TXFR n).....	2171
42.4.11	Receive FIFO Registers (DSPI_RXFR n).....	2172
42.4.12	DSI Configuration Register 0 (DSPI_DSICR0).....	2173
42.4.13	DSI Serialization Data Register 0 (DSPI_SDR0).....	2175
42.4.14	DSI Alternate Serialization Data Register 0 (DSPI_ASDR0).....	2176
42.4.15	DSI Transmit Comparison Register 0 (DSPI_COMPR0).....	2176
42.4.16	DSI Deserialization Data Register 0 (DSPI_DDR0).....	2177
42.4.17	DSI Configuration Register 1 (DSPI_DSICR1).....	2177
42.4.18	DSI Serialization Source Select Register 0 (DSPI_SSR0).....	2179
42.4.19	DSI Parallel Input Select Register 0 (DSPI_PISR0).....	2179
42.4.20	DSI Parallel Input Select Register 1 (DSPI_PISR1).....	2180
42.4.21	DSI Parallel Input Select Register 2 (DSPI_PISR2).....	2181
42.4.22	DSI Parallel Input Select Register 3 (DSPI_PISR3).....	2182
42.4.23	DSI Deserialized Data Interrupt Mask Register 0 (DSPI_DIMR0).....	2183
42.4.24	DSI Deserialized Data Polarity Interrupt Register 0 (DSPI_DPIR0).....	2183
42.5	Functional description.....	2184
42.5.1	Start and Stop of module transfers.....	2185
42.5.2	Serial Peripheral Interface (SPI) configuration.....	2186
42.5.3	Deserial Serial Interface (DSI) Configuration.....	2190
42.5.4	Combined Serial Interface (CSI) Configuration.....	2195
42.5.5	Module baud rate and clock delay generation.....	2197
42.5.6	Transfer formats.....	2200
42.5.7	Continuous Serial Communications Clock.....	2210
42.5.8	Slave Mode Operation Constraints.....	2212

Section number	Title	Page
42.5.9	Timed Serial Bus (TSB).....	2212
42.5.10	Parity Generation and Check.....	2215
42.5.11	Interrupts/DMA requests.....	2217
42.5.12	Power saving features.....	2220
42.6	Initialization/application information.....	2221
42.6.1	How to manage queues.....	2221
42.6.2	Switching Master and Slave mode.....	2222
42.6.3	Initializing Module in Master/Slave Modes.....	2222
42.6.4	Baud rate settings.....	2222
42.6.5	Delay settings.....	2223
42.6.6	Calculation of FIFO pointer addresses.....	2224

Chapter 43 Zipwire

43.1	Chip-specific Zipwire information.....	2227
43.2	Overview.....	2227
43.3	Introduction.....	2228
43.4	Zipwire Block Diagram.....	2228
43.5	Architecture.....	2229
43.6	Zipwire interconnections.....	2230
43.7	Zipwire performance.....	2231
43.7.1	Read performance.....	2231
43.7.2	Write performance.....	2233

Chapter 44 Serial Interprocessor Interface (SIPI)

44.1	Chip-specific SIPI information.....	2237
44.1.1	SIPI Chip ID.....	2237
44.2	Introduction.....	2237
44.2.1	Scalability.....	2237
44.3	Overview.....	2238

Section number	Title	Page
44.4	SIPI block diagram.....	2240
44.5	Feature description.....	2240
44.5.1	Main features.....	2240
44.5.2	Standard features.....	2241
44.6	SIPI operation from reset.....	2241
44.7	Functional description.....	2241
44.7.1	External signals.....	2241
44.7.2	Frame format.....	2242
44.8	Transfer types.....	2248
44.8.1	Read transfer.....	2248
44.8.2	Register read answer transfer.....	2249
44.8.3	Register Write transfer.....	2250
44.8.4	Write Acknowledge transfer.....	2253
44.8.5	ID request response.....	2253
44.9	Transfer API and flow charts.....	2255
44.10	DMA programming sequence.....	2262
44.11	Modes of operation.....	2263
44.11.1	Initialization mode.....	2263
44.11.2	Normal mode.....	2263
44.11.3	Module Disable (MD).....	2263
44.12	Errors.....	2264
44.12.1	Timeout error.....	2264
44.12.2	CRC error.....	2264
44.12.3	Maximum count reached error.....	2265
44.12.4	Transaction ID error.....	2265
44.12.5	Acknowledge error.....	2265
44.13	CRC calculation.....	2265
44.14	Interrupt logic.....	2266
44.15	SIPI control and status overview.....	2267

Section number	Title	Page
44.16	Memory map and register definition.....	2268
44.16.1	SIPI Channel Control Register 0 (SIPI_CCR0).....	2271
44.16.2	SIPI Channel Status Register 0 (SIPI_CSR0).....	2274
44.16.3	SIPI Channel Interrupt Register 0 (SIPI_CIR0).....	2275
44.16.4	SIPI Channel Timeout Register 0 (SIPI_CTOR0).....	2276
44.16.5	SIPI Channel CRC Register 0 (SIPI_CCRC0).....	2277
44.16.6	SIPI Channel Address Register 0 (SIPI_CAR0).....	2277
44.16.7	SIPI Channel Data Register 0 (SIPI_CDR0).....	2278
44.16.8	SIPI Channel Control Register 1 (SIPI_CCR1).....	2278
44.16.9	SIPI Channel Status Register 1 (SIPI_CSR1).....	2281
44.16.10	SIPI Channel Interrupt Register 1 (SIPI_CIR1).....	2283
44.16.11	SIPI Channel Timeout Register 1 (SIPI_CTOR1).....	2284
44.16.12	SIPI Channel CRC Register 1 (SIPI_CCRC1).....	2285
44.16.13	SIPI Channel Address Register 1 (SIPI_CAR1).....	2285
44.16.14	SIPI Channel Data Register 1 (SIPI_CDR1).....	2286
44.16.15	SIPI Channel Control Register 2 (SIPI_CCR2).....	2286
44.16.16	SIPI Channel Status Register 2 (SIPI_CSR2).....	2289
44.16.17	SIPI Channel Interrupt Register 2 (SIPI_CIR2).....	2291
44.16.18	SIPI Channel Timeout Register 2 (SIPI_CTOR2).....	2292
44.16.19	SIPI Channel CRC Register 2 (SIPI_CCRC2).....	2293
44.16.20	SIPI Channel Address Register 2 (SIPI_CAR2).....	2293
44.16.21	SIPI Channel Data Register 2 (SIPI_CDR2_n).....	2294
44.16.22	SIPI Channel Control Register 3 (SIPI_CCR3).....	2294
44.16.23	SIPI Channel Status Register 3 (SIPI_CSR3).....	2297
44.16.24	SIPI Channel Interrupt Register 3 (SIPI_CIR3).....	2299
44.16.25	SIPI Channel Timeout Register 3 (SIPI_CTOR3).....	2300
44.16.26	SIPI Channel CRC Register 3 (SIPI_CCRC3).....	2301
44.16.27	SIPI Channel Address Register 3 (SIPI_CAR3).....	2301
44.16.28	SIPI Channel Data Register 3 (SIPI_CDR3).....	2302

Section number	Title	Page
44.16.29	SIPI Module Configuration Register (SIPI_MCR).....	2302
44.16.30	SIPI Status Register (SIPI_SR).....	2305
44.16.31	SIPI Max Count Register (SIPI_MAXCR).....	2307
44.16.32	SIPI Address Reload Register (SIPI_ARR).....	2307
44.16.33	SIPI Address Count Register (SIPI_ACR).....	2308
44.16.34	SIPI Error Register (SIPI_ERR).....	2309

Chapter 45

LVDS Fast Asynchronous Serial Transmission (LFAST) – Interprocessor Communications

45.1	Chip-specific LFAST information.....	2313
45.2	Introduction.....	2313
45.3	Block diagram	2313
45.4	External signals.....	2314
45.4.1	LFAST operating data rates.....	2314
45.5	LFAST frame structure.....	2315
45.6	Features.....	2318
45.7	Memory map and register definition.....	2319
45.7.1	LFAST Mode Configuration Register (LFAST_MCR).....	2321
45.7.2	LFAST Speed Control Register (LFAST_SCR).....	2323
45.7.3	LFAST Correlator Control Register (LFAST_COCR).....	2324
45.7.4	LFAST Test Mode Control Register (LFAST_TMCR).....	2326
45.7.5	LFAST Auto Loopback Control Register (LFAST_ALCR).....	2327
45.7.6	LFAST Rate Change Delay Control Register (LFAST_RCDCR).....	2328
45.7.7	LFAST Wakeup Delay Control Register (LFAST_SLCR).....	2328
45.7.8	LFAST ICLC Control Register (LFAST_ICR).....	2330
45.7.9	LFAST Ping Control Register (LFAST_PICR).....	2331
45.7.10	LFAST Rx FIFO CTS Control Register (LFAST_RFCR).....	2332
45.7.11	LFAST Tx Interrupt Enable Register (LFAST_TIER).....	2332
45.7.12	LFAST Rx Interrupt Enable Register (LFAST_RIER).....	2333
45.7.13	LFAST Rx ICLC Interrupt Enable Register (LFAST_RIIER).....	2335

Section number	Title	Page
45.7.14	LFAST PLL Control Register (LFAST_PLLCR).....	2337
45.7.15	LFAST LVDS Control Register (LFAST_LCR).....	2339
45.7.16	LFAST Unsolicited Tx Control Register (LFAST_UNSTCR).....	2342
45.7.17	LFAST Unsolicited Tx Data Registers (LFAST_UNSTDR _n).....	2342
45.7.18	LFAST Global Status Register (LFAST_GSR).....	2343
45.7.19	LFAST Ping Status Register (LFAST_PISR).....	2345
45.7.20	LFAST Data Frame Status Register (LFAST_DFSR).....	2345
45.7.21	LFAST Tx Interrupt Status Register (LFAST_TISR).....	2346
45.7.22	LFAST Rx Interrupt Status Register (LFAST_RISR).....	2347
45.7.23	LFAST Rx ICLC Interrupt Status Register (LFAST_RIISR).....	2349
45.7.24	LFAST PLL and LVDS Status Register (LFAST_PLLLSR).....	2351
45.7.25	LFAST Unsolicited Rx Status Register (LFAST_UNSRSR).....	2352
45.7.26	LFAST Unsolicited Rx Data Register (LFAST_UNSRDR _n).....	2353
45.8	Functional description.....	2353
45.8.1	Startup procedure.....	2353
45.8.2	Line Receiver.....	2357
45.8.3	Transmit Controller.....	2366
45.8.4	CTS mode support.....	2371
45.8.5	Frames supported.....	2372
45.8.6	Frame flow.....	2373
45.8.7	Test and Debug Support.....	2380
45.8.8	Interrupts.....	2389
45.9	Packet memory.....	2392
45.10	Resets.....	2393
45.11	Clocks.....	2394
45.11.1	Clocking strategy.....	2394
45.11.2	Slow speed clock.....	2395
45.11.3	Rx Controller Clocks.....	2398
45.11.4	Clocking Module Requirements for High Speed Phases.....	2398

Section number	Title	Page
45.11.5	Clock module requirements for low speed phases.....	2399
45.11.6	Tx Controller Clocks.....	2400

Chapter 46 Fast Ethernet Controller (FEC)

46.1	Chip-specific FEC information.....	2403
46.1.1	FEC interface selection.....	2403
46.1.2	FEC clocking.....	2403
46.2	Introduction.....	2403
46.2.1	Block diagram.....	2403
46.2.2	Features.....	2405
46.2.3	Clocking requirements.....	2406
46.3	Modes of operation.....	2406
46.3.1	Full and half duplex operation.....	2407
46.3.2	Interface options.....	2407
46.3.3	Address recognition options.....	2409
46.3.4	Internal loopback.....	2409
46.4	External signal description.....	2409
46.5	Memory map and register definition.....	2410
46.5.1	Interrupt Event Register (FEC_EIR).....	2415
46.5.2	Interrupt Mask Register (FEC_EIMR).....	2417
46.5.3	Receive Descriptor Active Register (FEC_RDAR).....	2419
46.5.4	Transmit Descriptor Active Register (FEC_TDAR).....	2420
46.5.5	Ethernet Control Register (FEC_ECR).....	2421
46.5.6	MII Management Frame Register (FEC_MMFR).....	2422
46.5.7	MII Speed Control Register (FEC_MSCR).....	2423
46.5.8	MIB Control Register (FEC_MIBC).....	2424
46.5.9	Receive Control Register (FEC_RCR).....	2426
46.5.10	Transmit Control Register (FEC_TCR).....	2428
46.5.11	Physical Address Low Register (FEC_PALR).....	2429

Section number	Title	Page
46.5.12	Physical Address High Register and Type Field (FEC_PAUR).....	2430
46.5.13	Opcode/Pause Duration (FEC_OPD).....	2430
46.5.14	Descriptor Individual Upper Address Register (FEC_IAUR).....	2431
46.5.15	Descriptor Individual Lower Address Register (FEC_IALR).....	2431
46.5.16	Descriptor Group Upper Address Register (FEC_GAUR).....	2432
46.5.17	Descriptor Group Lower Address Register (FEC_GALR).....	2432
46.5.18	Transmit FIFO Watermark (FEC_TFWR).....	2433
46.5.19	FIFO Receive Bound Register (FEC_FRBR).....	2434
46.5.20	FIFO Receive Start Register (FEC_FRSR).....	2434
46.5.21	Receive Descriptor Ring Start Register (FEC_ERDSR).....	2435
46.5.22	Transmit Buffer Descriptor Ring Start Register (FEC_ETDSR).....	2436
46.5.23	Receive Buffer Size Register (FEC_EMRBR).....	2436
46.5.24	Count of frames not counted correctly (FEC_RMON_T_DROP).....	2437
46.5.25	RMON Tx packet count (FEC_RMON_T_PACKETS).....	2438
46.5.26	RMON Tx broadcast packets (FEC_RMON_T_BC_PKT).....	2438
46.5.27	RMON Tx multicast packets (FEC_RMON_T_MC_PKT).....	2439
46.5.28	RMON Tx packets with CRC/align error (FEC_RMON_T_CRC_ALIGN).....	2439
46.5.29	RMON Tx packets < 64 bytes, good CRC (FEC_RMON_T_UNDERSIZE).....	2440
46.5.30	RMON Tx packets > MAX_FL bytes, good CRC (FEC_RMON_T_OVERSIZE).....	2440
46.5.31	RMON Tx packets < 64 bytes, bad CRC (FEC_RMON_T_FRAG).....	2441
46.5.32	RMON Tx packets > MAX_FL bytes, bad CRC (FEC_RMON_T_JAB).....	2441
46.5.33	RMON Tx collision count (FEC_RMON_T_COL).....	2442
46.5.34	RMON Tx 64 byte packets (FEC_RMON_T_P64).....	2442
46.5.35	RMON Tx 65 to 127 byte packets (FEC_RMON_T_P65TO127).....	2443
46.5.36	RMON Tx 128 to 255 byte packets (FEC_RMON_T_P128TO255).....	2443
46.5.37	RMON Tx 256 to 511 byte packets (FEC_RMON_T_P256TO511).....	2444
46.5.38	RMON Tx 512 to 1023 byte packets (FEC_RMON_T_P512TO1023).....	2444
46.5.39	RMON Tx 1024 to 2047 byte packets (FEC_RMON_T_P1024TO2047).....	2445
46.5.40	RMON Tx packets with > 2048 bytes (FEC_RMON_T_P_GTE2048).....	2445

Section number	Title	Page
46.5.41	RMON Tx Octets (FEC_RMON_T_OCTETS).....	2446
46.5.42	Count of transmitted frames not counted correctly (FEC_IEEE_T_DROP).....	2446
46.5.43	Frames transmitted OK (FEC_IEEE_T_FRAME_OK).....	2447
46.5.44	Frames transmitted with single collision (FEC_IEEE_T_ICOL).....	2447
46.5.45	Frames transmitted with multiple collisions (FEC_IEEE_T_MCOL).....	2448
46.5.46	Frames transmitted after deferral delay (FEC_IEEE_T_DEF).....	2448
46.5.47	Frames transmitted with late collision (FEC_IEEE_T_LCOL).....	2449
46.5.48	Frames transmitted with excessive collisions (FEC_IEEE_T_EXCOL).....	2449
46.5.49	Frames transmitted with Tx FIFO underrun (FEC_IEEE_T_MACERR).....	2450
46.5.50	Frames transmitted with carrier sense error (FEC_IEEE_T_CSERR).....	2450
46.5.51	Frames transmitted with SQE error (FEC_IEEE_T_SQE).....	2451
46.5.52	Flow control pause frames transmitted (FEC_IEEE_T_FDXFC).....	2451
46.5.53	Octet count for frames transmitted without error (FEC_IEEE_T_OCTETS_OK).....	2452
46.5.54	Count of received frames not counted correctly (FEC_RMON_R_DROP).....	2452
46.5.55	RMON Rx packet count (FEC_RMON_R_PACKETS).....	2453
46.5.56	RMON Rx broadcast packets (FEC_RMON_R_BC_PKT).....	2453
46.5.57	RMON Rx multicast packets (FEC_RMON_R_MC_PKT).....	2454
46.5.58	RMON Rx packets with CRC/Align error (FEC_RMON_R_CRC_ALIGN).....	2454
46.5.59	RMON Rx packets < 64 bytes, good CRC (FEC_RMON_R_UNDERSIZE).....	2455
46.5.60	RMON Rx packets > MAX_FL bytes, good CRC (FEC_RMON_R_OVERSIZE).....	2455
46.5.61	RMON Rx packets < 64 bytes, bad CRC (FEC_RMON_R_FRAG).....	2456
46.5.62	RMON Rx packets > MAX_FL bytes, bad CRC (FEC_RMON_R_JAB).....	2456
46.5.63	Reserved (FEC_RMON_R_RESVD_0).....	2457
46.5.64	RMON Rx 64 byte packets (FEC_RMON_R_P64).....	2457
46.5.65	RMON Rx 65 to 127 byte packets (FEC_RMON_R_P65TO127).....	2458
46.5.66	RMON Rx 128 to 255 byte packets (FEC_RMON_R_P128TO255).....	2458
46.5.67	RMON Rx 256 to 511 byte packets (FEC_RMON_R_P256TO511).....	2459
46.5.68	RMON Rx 512 to 1023 byte packets (FEC_RMON_R_P512TO1023).....	2459
46.5.69	RMON Rx 1024 to 2047 byte packets (FEC_RMON_R_P1024TO2047).....	2460

Section number	Title	Page
46.5.70	RMON Rx packets with > 2048 bytes (FEC_RMON_R_P_GTE2048).....	2460
46.5.71	RMON Rx octets (FEC_RMON_R_OCTETS).....	2461
46.5.72	Count of received frames not counted correctly (FEC_IEEE_R_DROP).....	2461
46.5.73	Frames received OK (FEC_IEEE_R_FRAME_OK).....	2462
46.5.74	Frames received with CRC error (FEC_IEEE_R_CRC).....	2462
46.5.75	Frames received with alignment error (FEC_IEEE_R_ALIGN).....	2463
46.5.76	Receive FIFO overflow count (FEC_IEEE_R_MACERR).....	2463
46.5.77	Flow control pause frames received (FEC_IEEE_R_FDXFC).....	2464
46.5.78	Octet count for frames received without error (FEC_IEEE_R_OCTETS_OK).....	2464
46.5.79	Using the RMON and IEEE registers.....	2464
46.6	Functional description.....	2465
46.6.1	MII data frame.....	2465
46.6.2	MII management frame structure.....	2466
46.6.3	Buffer descriptors.....	2467
46.6.4	Initialization sequence.....	2473
46.6.5	User initialization (Prior to Setting ECR[ETHER_EN]).....	2473
46.6.6	Microcontroller initialization.....	2474
46.6.7	User initialization (after setting ECR[ETHER_EN]).....	2475
46.6.8	Network interface options.....	2475
46.6.9	FEC frame transmission.....	2476
46.6.10	FEC frame reception.....	2478
46.6.11	Ethernet address recognition.....	2479
46.6.12	Hash algorithm.....	2482
46.6.13	Full duplex flow control.....	2485
46.6.14	Inter-Packet Gap (IPG) time.....	2486
46.6.15	Collision managing.....	2486
46.6.16	MII internal and external loopback.....	2486
46.6.17	RMII loopback.....	2487
46.6.18	RMII echo.....	2487

Section number	Title	Page
46.6.19	Ethernet error-managing procedure.....	2487

Chapter 47

Peripheral Sensor Interface (PSI5)

47.1	Chip-specific PSI5 information.....	2491
47.1.1	PSI5 GTM interface.....	2491
47.1.2	Detection of PSI5 SMC messages.....	2491
47.2	Introduction.....	2492
47.2.1	Overview.....	2492
47.2.2	Features.....	2493
47.2.3	Modes of operation.....	2495
47.3	External signal description.....	2499
47.4	Memory map and register description.....	2499
47.4.1	Global Control Register (PSI5_GCR).....	2505
47.4.2	PSI5 Channel Control Register (PSI5_CH0_PCCR).....	2507
47.4.3	DMA Control Register (PSI5_CH0_DCR).....	2513
47.4.4	DMA Status Register (PSI5_CH0_DSR).....	2517
47.4.5	General Interrupt Control Register (PSI5_CH0_GICR).....	2519
47.4.6	New Data Interrupt Control Register (PSI5_CH0_NDICR).....	2521
47.4.7	Overwrite Interrupt Control Register (PSI5_CH0_OWICR).....	2521
47.4.8	Error Interrupt Control Register (PSI5_CH0_EICR).....	2522
47.4.9	General Interrupt Status Register (PSI5_CH0_GISR).....	2523
47.4.10	DMA PSI5 Message Register (PSI5_CH0_DPMR).....	2525
47.4.11	DMA SMC Frame Register (PSI5_CH0_DSFR).....	2526
47.4.12	DMA Diagnostic Status Register (PSI5_CH0_DDSDR).....	2526
47.4.13	PSI5 Message Receive Register Low (PSI5_CH0_PMRRL).....	2527
47.4.14	PSI5 Message Receive Register High (PSI5_CH0_PMRRH).....	2527
47.4.15	PSI5 Message Register Low i (PSI5_CH0_PMRL <i>n</i>).....	2529
47.4.16	PSI5 Message Register High i (PSI5_CH0_PMRH <i>n</i>).....	2530
47.4.17	SMC Frame Register n (PSI5_CH0_SFR <i>n</i>).....	2531

Section number	Title	Page
47.4.18	New Data Status Register (PSI5_CH0_NDSR).....	2533
47.4.19	Overwrite Status Register (PSI5_CH0_OWSR).....	2533
47.4.20	Error Indication Status Register (PSI5_CH0_EISR).....	2534
47.4.21	Set New Data Status Register (PSI5_CH0_SNDSR).....	2534
47.4.22	Set Overwrite Status Register (PSI5_CH0_SOWSR).....	2535
47.4.23	Set Error Status Register (PSI5_CH0_SEISR).....	2535
47.4.24	Set SMC Error Status Register (PSI5_CH0_SSESr).....	2536
47.4.25	Sync Time Stamp Read Register (PSI5_CH0_STSRR).....	2537
47.4.26	Data Time Stamp Read Register (PSI5_CH0_DTSRR).....	2537
47.4.27	Slot n Frame Configuration Register (PSI5_CH0_SnFCR).....	2538
47.4.28	Slot 1 Start Boundary Register (PSI5_CH0_S1SBR).....	2540
47.4.29	Slot 2 Start Boundary Register (PSI5_CH0_S2SBR).....	2540
47.4.30	Slot 3 Start Boundary Register (PSI5_CH0_S3SBR).....	2541
47.4.31	Slot 4 Start Boundary Register (PSI5_CH0_S4SBR).....	2541
47.4.32	Slot 5 Start Boundary Register (PSI5_CH0_S5SBR).....	2542
47.4.33	Slot 6 Start Boundary Register (PSI5_CH0_S6SBR).....	2542
47.4.34	Slot n End Boundary Register (PSI5_CH0_SnEBR).....	2543
47.4.35	Data Output Block Configuration Register (PSI5_CH0_DOBCR).....	2544
47.4.36	Manchester Decoder Disable Offset (PSI5_CH0_MDDIS_OFF).....	2547
47.4.37	Pulse Width for Data Bit Value 0 (PSI5_CH0_PW0D).....	2548
47.4.38	Pulse Width for Data Bit Value 1 (PSI5_CH0_PW1D).....	2548
47.4.39	Counter Target Pulse Register (PSI5_CH0_CTPR).....	2549
47.4.40	Counter Initialize Pulse Register (PSI5_CH0_CIPR).....	2550
47.4.41	Data Preparation Register Low (PSI5_CH0_DPRL).....	2550
47.4.42	Data Preparation Register High (PSI5_CH0_DPRH).....	2551
47.4.43	Data Buffer Register Low (PSI5_CH0_DBRL).....	2552
47.4.44	Data Buffer Register High (PSI5_CH0_DBRH).....	2554
47.4.45	Data Shift Register Low (PSI5_CH0_DSRL).....	2556
47.4.46	Data Shift Register High (PSI5_CH0_DSRH).....	2557

Section number	Title	Page
47.4.47	Device modes and Register bit accesses.....	2557
47.5	Functional description.....	2559
47.5.1	Sensor-to-ECU communication.....	2559
47.5.2	Data Link Layer.....	2560
47.5.3	ECU-to-sensor communication.....	2582
47.5.4	Tx/Rx scenario.....	2594
47.5.5	Interrupt handling.....	2594
47.5.6	DMA support.....	2597
47.5.7	Message read through interrupt generation.....	2600
47.6	Initialization information.....	2600
47.6.1	Initialization.....	2600
47.7	Design Assumptions and Limitations.....	2601
47.7.1	Assumptions.....	2601
47.7.2	Limitations.....	2602
47.8	Miscellaneous descriptions.....	2602
47.8.1	Internal SYNC Pulse generation and coordination across PSI5 channels.....	2602
47.8.2	Valid states for integrated sync pulse generator.....	2606
47.8.3	Implementation of EI Diagnostic Register.....	2609

Chapter 48 SENT Receiver (SRX)

48.1	Chip-specific SRX information.....	2611
48.2	Introduction.....	2611
48.2.1	Features.....	2612
48.2.2	Modes of operation.....	2613
48.2.3	Block diagram.....	2615
48.2.4	Design overview.....	2617
48.3	External signal description.....	2618
48.3.1	SENT_RX [(CH-1):0].....	2618

Section number	Title	Page
48.4	Memory map and register definition.....	2619
48.4.1	Global Control Register (SRX_GBL_CTRL).....	2623
48.4.2	Channel Enable Register (SRX_CHNL_EN).....	2625
48.4.3	Global Status Register (SRX_GBL_STATUS).....	2626
48.4.4	Fast Message Ready Status Register (SRX_FMSG_RDY).....	2627
48.4.5	Slow Serial Message Ready Status Register (SRX_SMSG_RDY).....	2628
48.4.6	Data Control Register 1 (SRX_DATA_CTRL1).....	2629
48.4.7	Fast Message DMA Control Register (SRX_FDMA_CTRL).....	2631
48.4.8	Slow Serial Message DMA Control Register (SRX_SDMA_CTRL).....	2631
48.4.9	Fast Message Ready Interrupt Control Register (SRX_FRDY_IE).....	2632
48.4.10	Slow Serial Message Ready Interrupt Enable Register (SRX_SRDY_IE).....	2633
48.4.11	DMA Fast Message Data Read Register (SRX_DMA_FMSG_DATA).....	2633
48.4.12	DMA Fast Message CRC Read Register (SRX_DMA_FMSG_CRC).....	2634
48.4.13	DMA Fast Message Time Stamp Read Register (SRX_DMA_FMSG_TS).....	2635
48.4.14	DMA Slow Serial Message Bit3 Read Register (SRX_DMA_SMSG_BIT3).....	2635
48.4.15	DMA Slow Serial Message Bit2 Read Register (SRX_DMA_SMSG_BIT2).....	2637
48.4.16	DMA Slow Serial Message Time Stamp Read Register (SRX_DMA_SMSG_TS).....	2637
48.4.17	Channel 'n' Clock Control Register (n = 0 to (CH-1)) (SRX_CHn_CLK_CTRL).....	2638
48.4.18	Channel 'n' Status Register (n=0 to (CH-1)) (SRX_CHn_STATUS).....	2640
48.4.19	Channel 'n' Configuration Register (n=0 to (CH-1)) (SRX_CHn_CONFIG).....	2643
48.4.20	Channel 'n' Fast Message Data Read Register (n=0 to (CH-1)) (SRX_CHn_FMSG_DATA).....	2646
48.4.21	Channel 'n' Fast Message CRC Read Register (n=0 to (CH-1)) (SRX_CHn_FMSG_CRC).....	2647
48.4.22	Channel 'n' Fast Message Time Stamp Read Register (n=0 to (CH-1)) (SRX_CHn_FMSG_TS).....	2647
48.4.23	Channel 'n' Serial Message Read Register (Bit 3) (n=0 to (CH-1)) (SRX_CHn_SMSG_BIT3).....	2648
48.4.24	Channel 'n' Serial Message Read Register (Bit 2)(n=0 to (CH-1)) (SRX_CHn_SMSG_BIT2).....	2649
48.4.25	Channel 'n' Serial Message Time Stamp Read Register (n=0 to (CH-1)) (SRX_CHn_SMSG_TS).....	2650
48.5	Functional description.....	2650
48.5.1	Initialization sequence.....	2650

Section number	Title	Page
48.5.2	DMA read logic.....	2651
48.5.3	Message reading via interrupts.....	2659
48.5.4	Overflow behavior.....	2661
48.5.5	Adjustment for variation in sensor (Tx) clock.....	2662
48.5.6	Input programmable filter.....	2663
48.5.7	Receiver diagnostics.....	2663
48.5.8	Time stamp logic.....	2669
48.5.9	Bus Idle Diagnostic.....	2670
48.6	Clocks and resets.....	2670
48.6.1	Clocking strategy.....	2670
48.6.2	System bus clock requirements.....	2671
48.6.3	High frequency receiver clock (protocol clock) requirements.....	2672

Chapter 49

Enhanced Serial Communication Interface (eSCI)

49.1	Chip-specific eSCI information.....	2675
49.1.1	eSCI clocking.....	2675
49.1.2	eSCI registers: core access in supervisor mode.....	2675
49.2	Introduction.....	2676
49.2.1	Bibliography.....	2676
49.2.2	Acronyms and Abbreviations.....	2676
49.2.3	Glossary.....	2676
49.2.4	Overview.....	2677
49.2.5	Features.....	2678
49.2.6	Modes of Operation.....	2680
49.3	External Signal Description.....	2681
49.3.1	Detailed Signal Descriptions.....	2681
49.4	Memory Map and Register Definition.....	2681
49.4.1	Baud Rate Register (eSCI_BRR).....	2682
49.4.2	Control Register 1 (eSCI_CR1).....	2683

Section number	Title	Page
49.4.3	Control Register 2 (eSCI_CR2).....	2685
49.4.4	SCI Data Register (eSCI_SDR).....	2687
49.4.5	Interrupt Flag and Status Register 1 (eSCI_IFSR1).....	2689
49.4.6	Interrupt Flag and Status Register 2 (eSCI_IFSR2).....	2691
49.4.7	LIN Control Register 1 (eSCI_LCR1).....	2692
49.4.8	LIN Control Register 2 (eSCI_LCR2).....	2694
49.4.9	LIN Transmit Register (eSCI_LTR).....	2694
49.4.10	LIN Receive Register (eSCI_LRR).....	2696
49.4.11	LIN CRC Polynomial Register (eSCI_LPR).....	2697
49.4.12	Control Register 3 (eSCI_CR3).....	2697
49.5	Functional Description.....	2698
49.5.1	Module Control.....	2698
49.5.2	Frame Formats.....	2698
49.5.3	Baud Rate and Clock Generation.....	2702
49.5.4	Baud Rate Tolerance.....	2704
49.5.5	SCI Mode.....	2706
49.5.6	LIN Mode.....	2723
49.5.7	Interrupts.....	2734
49.6	Application Information.....	2735
49.6.1	SCI Data Frames Separated by Preamble.....	2735

Chapter 50

Resets

50.1	Overview.....	2737
50.2	Reset sequence.....	2738
50.3	Reset pins.....	2741
50.3.1	RESET.....	2741
50.3.2	RSTOUT.....	2741

Section number	Title	Page
50.4	Reset sources.....	2742
50.4.1	Power-on reset.....	2742
50.4.2	External reset.....	2744
50.4.3	FCCU reset.....	2744
50.4.4	FOSU reset.....	2745
50.4.5	Dual core reset.....	2745
50.4.6	JTAG reset.....	2745
50.4.7	Software system reset.....	2746
50.4.8	Software external reset.....	2746
50.4.9	STCU reset.....	2746
50.4.10	Life cycle manipulation reset.....	2747
50.5	Reset configuration.....	2747
50.5.1	Weak pull up/down configuration.....	2747
50.5.2	Boot configuration.....	2748
50.6	Boot sequence.....	2748

Chapter 51 System Boot

51.1	Overview.....	2751
51.2	Features.....	2751
51.3	Modes of operation.....	2752
51.3.1	Normal boot.....	2752
51.3.2	Parallel secure boot.....	2752
51.3.3	Debug mode.....	2752
51.3.4	Internal boot mode.....	2752
51.3.5	Serial boot mode.....	2753
51.3.6	Development bus boot mode.....	2753
51.4	Memory Map.....	2753
51.5	Functional Description.....	2754
51.5.1	BAM program flow chart.....	2754

Section number	Title	Page
51.5.2	BAM program operation.....	2754
51.5.3	Reset Configuration Half Word (RCHW).....	2756
51.5.4	Internal boot mode.....	2758
51.5.5	Serial boot mode.....	2759
51.5.6	Booting from the development bus.....	2768

Chapter 52

System Status and Configuration Module (SSCM)

52.1	Chip-specific SSCM information.....	2771
52.1.1	SSCM and UTEST flash memory.....	2771
52.1.2	SSCM security reset and LC state.....	2772
52.1.3	SSCM_ERROR[PAE] and abort responses to illegal accesses.....	2773
52.2	Introduction.....	2773
52.2.1	Overview.....	2773
52.2.2	Glossary.....	2774
52.2.3	Features.....	2774
52.2.4	Modes of operation.....	2775
52.3	External signal description.....	2775
52.4	Memory map and register definition.....	2775
52.4.1	SSCM System Status (SSCM_STATUS).....	2776
52.4.2	SSCM Error Configuration Register (SSCM_ERROR).....	2778
52.4.3	Life Cycle Status Register (SSCM_LCSTAT).....	2779
52.5	Functional description.....	2779
52.5.1	DCF mechanism.....	2780
52.5.2	Life Cycle.....	2782
52.6	Initialization and application information.....	2784
52.6.1	Reset.....	2784
52.7	Additional safety measures.....	2784
52.7.1	Spurious reset protection.....	2784

Section number	Title	Page
Chapter 53		
Power Management		
53.1	Overview.....	2785
53.2	Functional description.....	2786
53.3	Power modes.....	2786
53.4	External supply.....	2786
53.5	Power management supply description.....	2786
53.6	Voltage monitoring.....	2787

Chapter 54		
Power Management Controller (PMC)		
54.1	Introduction.....	2789
54.1.1	Features.....	2789
54.2	Memory Map and Registers.....	2791
54.2.1	Temperature Sensor Interrupt Enable Register (PMC_TS_IER).....	2792
54.2.2	Temperature Event Status Register (PMC_ESR_TD).....	2794
54.2.3	Temperature Reset Event Enable register (PMC_REE_TD).....	2796
54.2.4	Temperature Reset Event Selection Register (PMC_RES_TD).....	2798
54.2.5	Temperature Detector Configuration Register (PMC_CTL_TD).....	2799
54.2.6	Temperature Sensor Fault Injection Register (PMC_TS_FIR).....	2800
54.2.7	Voltage Detect User Mode Test Register (PMC_SELF_TEST_UM_VD_REG).....	2801
54.2.8	FCCU Fault Injection Register (PMC_FIR).....	2804
54.2.9	LVD/HVD Interrupt Enable Control Register (PMC_LVD_HVD_INTR_ENB).....	2805
54.2.10	LVD/HVD Event Status Register (PMC_LVD_HVD_EVENT_STATUS).....	2808
54.2.11	ADC Channel Select Register (PMC_ADC_CS).....	2810
54.2.12	Reset Event Enable Register (PMC_REE).....	2812
54.3	Functional description.....	2814
54.3.1	Analog PMC interface.....	2814
54.3.2	Temperature Sensor interface logic.....	2814
54.3.3	POR phase gates.....	2814

Section number	Title	Page
54.3.4	Flash memory interface to DCF.....	2814
54.3.5	FCCU interface.....	2815
54.3.6	LVD/HVD self test mechanism.....	2815

Chapter 55 JTAG Controller (JTAGC)

55.1	Chip-specific JTAGC information.....	2819
55.1.1	JTAG ID.....	2819
55.2	Introduction.....	2819
55.2.1	Block diagram.....	2819
55.2.2	Features.....	2820
55.2.3	Modes of operation.....	2821
55.3	External signal description.....	2822
55.3.1	TCK—Test clock input.....	2822
55.3.2	TDI—Test data input.....	2822
55.3.3	TDO—Test data output.....	2823
55.3.4	TMS—Test mode select.....	2823
55.3.5	JCOMP—JTAG compliancy.....	2823
55.4	Register description.....	2823
55.4.1	Instruction register.....	2823
55.4.2	Bypass register.....	2824
55.4.3	Device identification register.....	2824
55.4.4	JTAG_PASSWORD register.....	2825
55.4.5	Boundary scan register.....	2825
55.5	Functional description.....	2826
55.5.1	JTAGC reset configuration.....	2826
55.5.2	IEEE 1149.1-2001 (JTAG) Test Access Port.....	2826
55.5.3	TAP controller state machine.....	2826
55.5.4	JTAGC block instructions.....	2829
55.5.5	Boundary scan.....	2832

Section number	Title	Page
55.6	Initialization/Application information.....	2833
Chapter 56 IEEE 1149.7 Compact JTAG Test Access Port Controller (CJTAG)		
56.1	References.....	2835
56.2	Abbreviations.....	2835
56.3	Introduction.....	2836
56.3.1	Types of operation.....	2837
56.3.2	Deployment by class.....	2837
56.3.3	1149.7 TAP signals.....	2838
56.3.4	TAP.7 architecture.....	2839
56.3.5	Protocols.....	2840
56.4	Operating models.....	2842
56.5	CJTAG implementation summary.....	2842
56.5.1	T0 functions.....	2842
56.5.2	T1 functions.....	2843
56.5.3	T2 functions.....	2843
56.5.4	T3 functions.....	2843
56.5.5	T4 functions.....	2843
56.6	Ancillary services.....	2843
56.6.1	Overview.....	2843
56.6.2	Resets.....	2844
56.6.3	Start-up Options.....	2847
56.6.4	RSU operation.....	2848
56.6.5	TAPC State Machine.....	2850
56.7	EPU (Extended Protocol Unit) Operation.....	2851
56.7.1	EPU Operation.....	2851
56.7.2	EPU Registers.....	2854
56.7.3	EPU Commands.....	2866
56.7.4	EPU operating states.....	2874

Section number	Title	Page
56.7.5	System and EPU Paths.....	2875
56.8	APU (Advanced Protocol Unit) Operation.....	2875
56.8.1	Overview.....	2875
56.8.2	Operation.....	2878
56.8.3	Escape sequences.....	2880
56.8.4	Signal behaviors.....	2880
56.8.5	APU Functions.....	2881
56.8.6	Configuration Change Packets (CP).....	2887
56.8.7	Scan Packet.....	2888
56.8.8	SP Format.....	2888
56.9	Functional Description.....	2895
56.9.1	Switching from Standard Protocol to Advanced Protocol.....	2895

Chapter 57 JTAG Data Communication (JDC)

57.1	Introduction.....	2897
57.2	Overview.....	2897
57.3	Memory mapped registers.....	2898
57.3.1	Module Configuration Register (JDC_MCR).....	2899
57.3.2	Module Status Register (JDC_MSR).....	2900
57.3.3	JTAG Output Data Register (JDC_JOUT_IPS).....	2901
57.3.4	JTAG Input Data Register (JDC_JIN_IPS).....	2902
57.4	Non-Memory mapped register definition.....	2902
57.4.1	JTAG output data register (JOUT).....	2902
57.4.2	JTAG input data register (JIN).....	2903
57.5	Functional description.....	2903
57.5.1	JTAG register access.....	2904

Chapter 58 Nexus Development Interface (NDI)

58.1	Overview.....	2905

Section number	Title	Page
58.2	Nexus clients for TAP access.....	2906
58.3	Debug break and cross-triggering.....	2906
58.3.1	Break control.....	2907
58.3.2	Cross-triggering control.....	2907
58.3.3	Synchronous restart control.....	2907
58.4	Nexus eDMA Interface (NXDM) and Nexus SIPI Interface (NXSI).....	2908
58.4.1	Introduction.....	2908
58.4.2	External Signal Description.....	2910
58.4.3	NXDM and NXSI Programmers Model.....	2910
58.4.4	Functional Description.....	2921

Chapter 59 Nexus Port Controller (NPC)

59.1	Chip-specific NPC information.....	2931
59.1.1	DID register reset value.....	2931
59.1.2	NPC MDO pins.....	2931
59.2	Introduction.....	2932
59.2.1	Overview.....	2932
59.2.2	Features.....	2932
59.2.3	Modes of operation.....	2933
59.3	External signal description.....	2935
59.3.1	Overview.....	2935
59.3.2	Detailed signal descriptions.....	2935
59.4	Register definition.....	2937
59.4.1	Register descriptions.....	2937
59.5	Functional Description.....	2942
59.5.1	NPC reset configuration.....	2942
59.5.2	Auxiliary Output Port.....	2942
59.5.3	IEEE 1149.1-2001 (JTAG) TAP.....	2945
59.5.4	Nexus JTAG Port Sharing.....	2950

Section number	Title	Page
59.5.5	MCKO Control	2950
59.5.6	EVTO Sharing.....	2951
59.5.7	Nexus Reset Control.....	2951
59.5.8	System Clock Locked Indication.....	2951
59.6	Initialization/Application Information.....	2952
59.6.1	Accessing NPC tool-mapped registers.....	2952

Chapter 60

Nexus eTPU Development Interface (NDEDI)

60.1	Chip-specific NDEDI information.....	2953
60.1.1	NDEDI/NSEDI device-specific parameters.....	2953
60.2	Introduction.....	2954
60.3	Features.....	2955
60.4	Modes of operation.....	2957
60.4.1	Reset.....	2957
60.4.2	Disabled-port mode.....	2957
60.4.3	Full-port mode.....	2957
60.4.4	Reduced-port mode.....	2958
60.4.5	Stop mode.....	2958
60.5	External signal description.....	2958
60.5.1	Overview.....	2958
60.5.2	Detailed signal descriptions.....	2959
60.6	Register definition.....	2961
60.6.1	Register descriptions.....	2963
60.6.2	Unimplemented Registers.....	2996
60.7	Functional description.....	2997
60.7.1	NDEDI reset configuration.....	2997
60.7.2	Auxiliary output port.....	2997
60.7.3	IEEE 1149.1 (JTAG) Input Port.....	3007
60.7.4	Microcode Development Support.....	3012

Section number	Title	Page
60.7.5	Debug Status.....	3013
60.7.6	Ownership Trace.....	3015
60.7.7	Program Trace.....	3016
60.7.8	Data Trace.....	3031
60.7.9	Watchpoint Trace.....	3038
60.7.10	ETPU Message Queue.....	3040
60.8	Initialization/Application Information.....	3044
60.8.1	Accessing NDEDI tool-mapped registers.....	3044
60.8.2	Program Trace Reconstruction.....	3045
60.8.3	Microcode Development Support.....	3049

Chapter 61 Development Trigger Semaphore (DTS)

61.1	Introduction.....	3053
61.2	Overview.....	3053
61.3	DTS device connections.....	3055
61.3.1	DTS register access.....	3056
61.4	Memory mapped registers.....	3057
61.4.1	Output Enable Register (DTS_ENABLE).....	3057
61.4.2	Startup Register (DTS_STARTUP).....	3058
61.4.3	Semaphore Register (DTS_SEMAPHORE).....	3059
61.4.4	Semaphore Extension (DTS_SEMAPHORE_B).....	3060
61.5	Example application.....	3060

Chapter 62 Fault Collection and Control Unit (FCCU)

62.1	Chip-specific FCCU information.....	3063
62.1.1	FCCU register default values.....	3063
62.1.2	FOSU_COUNT value.....	3063
62.1.3	FCCU fault inputs.....	3064
62.1.4	Error signal flow diagram.....	3068

Section number	Title	Page
62.1.5	Restriction on configuring RESET and NMI reactions on same FCCU channel.....	3069
62.1.6	FCCU_CFG register event bit values by source (N and C).....	3070
62.2	Introduction.....	3071
62.2.1	Acronyms and abbreviations.....	3072
62.3	Main features.....	3073
62.4	Block diagram.....	3073
62.5	Signal description.....	3075
62.5.1	Signals.....	3075
62.6	Put FCCU in Configuration or Normal state.....	3077
62.6.1	Introduction.....	3077
62.6.2	About changing states.....	3077
62.6.3	Put FCCU in Configuration state.....	3077
62.6.4	Put FCCU in Normal state.....	3078
62.7	Run operations.....	3078
62.7.1	Introduction.....	3078
62.7.2	About running operations.....	3079
62.7.3	Run an operation.....	3079
62.8	Lock and unlock the configuration.....	3079
62.8.1	Introduction.....	3079
62.8.2	About locking the configuration.....	3080
62.8.3	Temporarily lock the configuration.....	3080
62.8.4	Temporarily unlock the configuration.....	3080
62.8.5	Permanently lock the configuration.....	3080
62.9	Functional description.....	3080
62.9.1	Definitions.....	3080
62.9.2	FSM description.....	3081
62.9.3	Fault priority scheme and nesting.....	3083
62.9.4	STCU interface.....	3084
62.9.5	Nonvolatile memory interface.....	3085

Section number	Title	Page
62.9.6	EOUT interface.....	3086
62.9.7	Error signal flow.....	3093
62.10	Register descriptions.....	3093
62.10.1	Control (FCCU_CTRL).....	3096
62.10.2	Control Key (FCCU_CTRLK).....	3098
62.10.3	Configuration (FCCU_CFG).....	3099
62.10.4	Noncritical Fault Configuration (FCCU_NCF_CFGn).....	3102
62.10.5	Noncritical Fault State Configuration (FCCU_NCFS_CFGn).....	3103
62.10.6	Noncritical Fault Status (FCCU_NCF_Sn).....	3104
62.10.7	Noncritical Fault Key (FCCU_NCFK).....	3106
62.10.8	Noncritical Fault Enable (FCCU_NCF_En).....	3107
62.10.9	Noncritical Fault Timeout Enable (FCCU_NCF_TOEn).....	3108
62.10.10	Noncritical Fault Timeout (FCCU_NCF_TO).....	3109
62.10.11	Configuration-State Timer Interval (FCCU_CFG_TO).....	3109
62.10.12	IO Control (FCCU_EINOUT).....	3110
62.10.13	Status (FCCU_STAT).....	3112
62.10.14	NA Freeze Status (FCCU_N2AF_STATUS).....	3114
62.10.15	AF Freeze Status (FCCU_A2FF_STATUS).....	3115
62.10.16	NF Freeze Status (FCCU_N2FF_STATUS).....	3116
62.10.17	FA Freeze Status (FCCU_F2A_STATUS).....	3117
62.10.18	Noncritical Fault Fake (FCCU_NCFK).....	3118
62.10.19	IRQ Status (FCCU_IRQ_STAT).....	3119
62.10.20	IRQ Enable (FCCU_IRQ_EN).....	3120
62.10.21	XTMR (FCCU_XTMR).....	3121
62.10.22	Transient Configuration Lock (FCCU_TRANS_LOCK).....	3122
62.10.23	Permanent Configuration Lock (FCCU_PERMNT_LOCK).....	3123
62.10.24	Delta T (FCCU_DELTA_T).....	3123
62.10.25	IRQ Alarm Enable (FCCU_IRQ_ALARM_ENn).....	3124
62.10.26	NMI Enable (FCCU_NMI_ENn).....	3125

Section number	Title	Page
62.10.27	Bistable Fault-Output (EOUT) Mode Signaling Enable (FCCU_EOUT_SIG_EN n).....	3126
62.10.28	Configuration registers.....	3127
62.10.29	FCCU_CFG register bit value sources (N and C) by event.....	3128
62.11	FCCU Output Supervision Unit.....	3129
62.12	Use cases and limitations.....	3131
62.12.1	Configuration guidelines.....	3131
62.12.2	Recommendations to configure FCCU.....	3132

Chapter 63

Self-Test Control Unit (STCU2)

63.1	Chip-specific STCU information.....	3135
63.1.1	Supported BIST sequences.....	3135
63.1.2	Configure or bypass STCU2 off-line self-tests.....	3136
63.1.3	STCU interfaces to other modules.....	3136
63.1.4	On-line LBIST and MBIST clock configuration.....	3137
63.1.5	STCU2_PLL_CFG configuration of PLLDIG_PLL0DV.....	3138
63.1.6	Wait time for writing to the on-line registers.....	3138
63.1.7	On-line MBIST.....	3138
63.1.8	LBIST scenarios.....	3141
63.1.9	STCU register default values.....	3145
63.1.10	AUTOLOCK_VALUE for register write access and DCF_COMPLETION value for DCF write completion.....	3146
63.2	Introduction.....	3146
63.3	Main features.....	3147
63.4	Block diagram.....	3149
63.5	IPS bus interface.....	3150
63.6	BISTs and BIST partitions.....	3151
63.6.1	Definition: BIST.....	3151
63.6.2	Definition: BIST partition.....	3151
63.6.3	Example: BIST partitions on a chip.....	3152

Section number	Title	Page
63.6.4	Types of BIST partitions.....	3152
63.7	BIST sequences.....	3152
63.7.1	Definition: BIST sequence.....	3152
63.7.2	STCU2 executes MBISTs before LBISTs.....	3152
63.7.3	Example: Single-phase BIST sequence.....	3153
63.7.4	Example: Multiphase BIST sequence.....	3153
63.7.5	Types of BIST sequences.....	3153
63.7.6	Supported BIST sequences.....	3154
63.7.7	Definition: Default offline BIST sequence.....	3154
63.8	Functional description.....	3154
63.8.1	FSM description.....	3154
63.8.2	Reset management.....	3155
63.8.3	Built-in self-test scheduling.....	3155
63.8.4	PLL interface.....	3156
63.8.5	Interrupt interface.....	3156
63.8.6	FCCU interface.....	3157
63.8.7	Watchdogs.....	3157
63.8.8	SSCM interface.....	3159
63.9	Register description.....	3159
63.9.1	STCU2 Run Register (STCU2_RUN).....	3166
63.9.2	STCU2 Run Software Register (STCU2_RUNSW).....	3167
63.9.3	STCU2 SK Code Register (STCU2_SKC).....	3169
63.9.4	STCU2 Configuration Register (STCU2_CFG).....	3170
63.9.5	STCU2 PLL Configuration Register (STCU2_PLL_CFG).....	3173
63.9.6	STCU2 Watchdog Granularity (STCU2_WDG).....	3174
63.9.7	STCU2 Interrupt Flag Register (STCU2_INT_FLG).....	3176
63.9.8	STCU2 Error Register (STCU2_ERR_STAT).....	3177
63.9.9	STCU2 Error FM Register (STCU2_ERR_FM).....	3180
63.9.10	STCU2 Off-Line LBIST Status Register (STCU2_LBS).....	3181

Section number	Title	Page
63.9.11	STCU2 Off-Line LBIST End Flag Register (STCU2_LBE).....	3182
63.9.12	STCU2 On-Line LBIST Status Register (STCU2_LBSSW).....	3183
63.9.13	STCU2 On-Line LBIST End Flag Register (STCU2_LBESW).....	3185
63.9.14	STCU2 On-Line LBIST Reset Management (STCU2_LBRMSW).....	3186
63.9.15	STCU2 LBIST Unrecoverable FM Register (STCU2_LBUFM).....	3188
63.9.16	STCU2 Off-Line MBIST Status Low Register (STCU2_MBSL).....	3189
63.9.17	STCU2 Off-Line MBIST Status Medium Register (STCU2_MBSM).....	3194
63.9.18	STCU2 Off-Line MBIST Status High Register (STCU2_MBSH).....	3199
63.9.19	STCU2 Off-Line MBIST End Flag Low Register (STCU2_MBEL).....	3200
63.9.20	STCU2 Off-Line MBIST End Flag Medium Register (STCU2_MBEM).....	3204
63.9.21	STCU2 Off-Line MBIST End Flag High Register (STCU2_MBEH).....	3208
63.9.22	STCU2 On-Line MBIST Status Low Register (STCU2_MBSLSW).....	3209
63.9.23	STCU2 On-Line MBIST Status Medium Register (STCU2_MBSMSW).....	3212
63.9.24	STCU2 On-Line MBIST Status High Register (STCU2_MBSHSW).....	3216
63.9.25	STCU2 On-Line MBIST End Flag Low Register (STCU2_MBELSW).....	3217
63.9.26	STCU2 On-Line MBIST End Flag Medium Register (STCU2_MBEMSW).....	3221
63.9.27	STCU2 On-Line MBIST End Flag High Register (STCU2_MBEHSW).....	3225
63.9.28	STCU2 MBIST Unrecoverable FM Low Register (STCU2_MBUFML).....	3226
63.9.29	STCU2 MBIST Unrecoverable FM Medium Register (STCU2_MBUFMM).....	3229
63.9.30	STCU2 MBIST Unrecoverable FM High Register (STCU2_MBUFMH).....	3233
63.9.31	STCU2 LBIST Control (STCU2_LB_CTRL n).....	3234
63.9.32	STCU2 LBIST PC Stop Register (STCU2_LB_PCS n).....	3237
63.9.33	STCU2 Off-Line LBIST MISR Expected Low Register (STCU2_LB_MISREL n).....	3238
63.9.34	STCU2 Off-Line LBIST MISR Expected High Register (STCU2_LB_MISREH n).....	3238
63.9.35	STCU2 Off-Line LBIST MISR Read Low Register (STCU2_LB_MISRRL n).....	3239
63.9.36	STCU2 Off-Line LBIST MISR Read High Register (STCU2_LB_MISRRH n).....	3240
63.9.37	STCU2 On-Line LBIST MISR Expected Low Register (STCU2_LB_MISRELSW n).....	3240
63.9.38	STCU2 On-Line LBIST MISR Expected High Register (STCU2_LB_MISREHSW n).....	3241
63.9.39	STCU2 On-Line LBIST MISR Read Low Register (STCU2_LB_MISRRLSW n).....	3242

Section number	Title	Page
63.9.40	STCU2 On-Line LBIST MISR Read High Register (STCU2_LB_MISRRHSW _n).....	3242
63.9.41	STCU2 MBIST Control Register (STCU2_MB_CTRL _n).....	3243
63.10	Use cases and limitations.....	3244
63.10.1	Offline self-test sequence.....	3244
63.10.2	Online self test sequence.....	3246
63.10.3	Bypass USER Mode.....	3248
63.10.4	Design implementation information.....	3248

Chapter 64 Cyclic Redundancy Check (CRC) Unit

64.1	Chip-specific CRC information.....	3251
64.2	Introduction.....	3251
64.3	Main features.....	3251
64.3.1	Standard features.....	3251
64.4	Block diagram.....	3252
64.5	External signal description.....	3252
64.5.1	Peripheral bus interface.....	3252
64.6	CRC memory map and registers.....	3253
64.6.1	Configuration Register (CRC_CFG _n).....	3254
64.6.2	Input Register (CRC_INP _n).....	3255
64.6.3	Current Status Register (CRC_CSTAT _n).....	3256
64.6.4	Output Register (CRC_OUTP _n).....	3257
64.7	Functional description.....	3257
64.8	Use cases.....	3259
64.8.1	Programming example.....	3259
64.8.2	Register programming.....	3261

Chapter 65 Security Overview

65.1	Introduction.....	3263
65.2	Security features summary.....	3263

Section number	Title	Page
65.3	Flash memory read, write, and erase protection.....	3264
Chapter 66 Cryptographic Services Engine (CSE)		
66.1	Chip-specific CSE information.....	3265
66.1.1	CSE configuration and SECURE_BOOT execution.....	3265
66.1.2	Sys Clk Freq for CSE.....	3265
66.1.3	CSE clock: minimum frequency.....	3265
66.1.4	CSE debug not supported.....	3266
66.1.5	CSE command-processing completion before Stop mode entry.....	3266
66.2	Introduction.....	3266
66.2.1	Overview.....	3266
66.2.2	Features.....	3266
66.2.3	Modes of operation.....	3267
66.2.4	Block diagram.....	3267
66.3	External signal description.....	3267
66.4	Memory Map and Register Definition.....	3268
66.4.1	CSE Control Register (CSE_CR).....	3268
66.4.2	CSE Status Register (CSE_SR).....	3270
66.4.3	CSE Interrupt Register (CSE_IR).....	3271
66.4.4	CSE Error Code Register (CSE_ECR).....	3272
66.4.5	CSE Command Register (CSE_CMD).....	3273
66.4.6	CSE Parameter Register (CSE_Pn).....	3274
66.5	CSE functional description.....	3274
66.5.1	Host interface.....	3274
66.5.2	Command processing.....	3275
66.5.3	Secure storage.....	3276
66.5.4	Encryption and decryption.....	3278
66.5.5	Message authentication.....	3278
66.5.6	Secure boot.....	3279

Section number	Title	Page
66.5.7	Random number generation.....	3280
66.5.8	Error handling.....	3280
66.6	CSE commands.....	3282
66.6.1	Encrypt ECB.....	3282
66.6.2	Encrypt CBC.....	3283
66.6.3	Decrypt ECB.....	3283
66.6.4	Decrypt CBC.....	3283
66.6.5	Generate MAC.....	3284
66.6.6	Verify MAC.....	3284
66.6.7	Load Key.....	3285
66.6.8	Load Plain Key.....	3286
66.6.9	Export RAM Key.....	3286
66.6.10	Initialize RNG.....	3286
66.6.11	Extend PRNG Seed.....	3287
66.6.12	Generate Random Number.....	3287
66.6.13	Secure Boot.....	3288
66.6.14	Boot Failure.....	3288
66.6.15	Boot OK.....	3288
66.6.16	Get ID.....	3289
66.6.17	Cancel.....	3289
66.6.18	Debug Challenge.....	3289
66.6.19	Debug Authorization.....	3290
66.6.20	Generate TRNG Random Number.....	3290
66.6.21	Initialize CSE.....	3291
66.6.22	Miyaguchi-Preneel (MP) Compression.....	3291

Chapter 67

Password and Device Security Module (PASS)

67.1	Chip-specific PASS information.....	3293
67.1.1	Accesses to unimplemented PASS registers.....	3293

Section number	Title	Page
67.1.2	PASS_LOCKm_PGn register bit mapping.....	3293
67.1.3	PASS chip select (client select).....	3297
67.1.4	Flash memory read protection scenarios.....	3297
67.1.5	Debug Interface Access.....	3298
67.1.6	Duration of Debug Interface Access via JTAG password.....	3299
67.1.7	Production Disable enablement.....	3299
67.1.8	PASS_LCSTAT[LIFE] value source.....	3300
67.2	Introduction.....	3300
67.2.1	Overview.....	3300
67.2.2	Features.....	3301
67.2.3	Modes of operation.....	3301
67.3	Memory map and register definition.....	3301
67.3.1	Life Cycle Status Register (PASS_LCSTAT).....	3303
67.3.2	Challenge Selector Register (PASS_CHSEL).....	3304
67.3.3	Challenge Status Register (PASS_CSTAT).....	3305
67.3.4	Challenge Input Register (PASS_CINn).....	3305
67.3.5	Password Group <i>n</i> - Lock 0 Status Register (PASS_LOCK0_PGn).....	3306
67.3.6	Password Group <i>n</i> - Lock 1 Status Register (PASS_LOCK1_PGn).....	3308
67.3.7	Password Group <i>n</i> - Lock 2 Status Register (PASS_LOCK2_PGn).....	3309
67.3.8	Password Group <i>n</i> - Lock 3 Status Register (PASS_LOCK3_PGn).....	3310
67.4	DCF clients.....	3313
67.4.1	Censorship.....	3314
67.4.2	Production Disable.....	3314
67.5	Functional description.....	3316
67.5.1	Censorship.....	3316
67.5.2	Debug Interface Access.....	3316
67.5.3	Flash Memory Read Protection.....	3317

Section number	Title	Page
67.5.4	Production Disable.....	3317
67.6	Initialization/application information.....	3318
67.6.1	Reset.....	3318
67.6.2	Setting lock bits in a password group.....	3318

Chapter 68 Tamper Detection Module (TDM)

68.1	Chip-specific TDM information.....	3321
68.1.1	Disabling the override of tamper detection.....	3321
68.1.2	TDRn_LOCKm DCF client bit mapping.....	3322
68.1.3	OTPENn DCF client bit mapping.....	3325
68.2	Overview.....	3328
68.3	Features.....	3329
68.4	External signal description.....	3329
68.5	DCF clients.....	3330
68.5.1	Diary Base Address (DBA) DCF client.....	3330
68.5.2	Tamper Region Override (TO) DCF client.....	3331
68.5.3	Software Tamper Region Override Disable (STO_DIS) DCF client.....	3332
68.5.4	One Time Programmable Enable (OTPEN0) DCF client.....	3333
68.5.5	One Time Programmable Enable (OTPEN1) DCF client.....	3334
68.5.6	One Time Programmable Enable (OTPEN2) DCF client.....	3334
68.5.7	One Time Programmable Enable (OTPEN3) DCF client.....	3335
68.5.8	Tamper Region Assignment DCF client (TDRn_LOCK0).....	3336
68.5.9	Tamper Region Assignment DCF client (TDRn_LOCK1).....	3337
68.5.10	Tamper Region Assignment DCF client (TDRn_LOCK2).....	3338
68.5.11	Tamper Region Assignment DCF client (TDRn_LOCK3).....	3339
68.6	Memory Map/Register Definition.....	3339
68.6.1	TDR Status Register (TDM_TDRSR).....	3340
68.6.2	Last Flash Programmed Address Register (TDM_LFPAR).....	3342
68.6.3	Diary Base Address (TDM_DBA).....	3343

Section number	Title	Page
68.6.4	Software Tamper Override Key Region (TDM_STO_KEY n).....	3343
68.7	Functional description.....	3344
68.7.1	Flash erase counter and tamper detection.....	3344
68.7.2	Assigning blocks to Tamper Detect Regions (TDR).....	3346
68.7.3	Diary.....	3346
68.7.4	Specifying the diary base address.....	3347
68.7.5	TDR lock status.....	3348
68.7.6	TDR override.....	3348
68.7.7	One time programmable (OTP).....	3349



Chapter 1

Preface

1.1 Overview

The primary objective of this document is to define the functionality of the MPC5777C Qorivva microcontroller for use by software and hardware developers. The MPC5777C is based on the Power Architecture® developed by Freescale/NXP.

The information in this document is preliminary and subject to change. As with any technical documentation, it is the reader's responsibility to ensure he or she is using the most recent version of the documentation.

To locate any published errata or updates for this document, visit <http://www.nxp.com>.

1.2 Device versions

This release of this document describes the functionality and programming model of maskset 3N45H.

With some exceptions, this document also describes maskset 2N45H. For information about 2N45H's differences from 3N45H, see the MPC5777C Reference Manual Addendum (document number MPC5777CRMAD).

1.3 Audience

This manual is intended for system software and hardware developers and applications programmers who want to develop products with the MPC5777C. It is assumed that the reader understands operating systems, microprocessor system design, basic principles of software and hardware, and basic details of the Power Architecture.

1.4 Document organization

This document includes two major sets of chapters.

Table 1-1. Chapter sets

Chapters or chapter groups	Description
Introduction Signal Description ¹ Memory Map, Interrupts, and DMA Requests Device Configuration Format (DCF) Records	Contain information that applies to all components on the chip
Core and system modules Clocking Memories and memory interfaces Analog measurement modules Timers Communication interfaces Reset and boot Power management Debug Safety modules Security modules	Detail particular areas of functionality, including individual modules on the chip

1. Details about external signal functions, their static electrical characteristics, and pad configuration settings for this chip appear in the spreadsheet that is embedded within this document: `MPC5777C_System_IO_Definition.xlsx`. To access this spreadsheet, view the document's list of attachments.

1.5 Conventions

1.5.1 Numbering systems

The following suffixes identify different numbering systems:

This suffix	Identifies a
b	Binary number. For example, the binary equivalent of the number 5 is written 101b. In some cases, binary numbers are shown with the prefix <i>0b</i> .
d	Decimal number. Decimal numbers are followed by this suffix only when the possibility of confusion exists. In general, decimal numbers are shown without a suffix.
h	Hexadecimal number. For example, the hexadecimal equivalent of the number 60 is written 3Ch. In some cases, hexadecimal numbers are shown with the prefix <i>0x</i> .

1.5.2 Typographic notation

The following typographic notation is used throughout this document:

Example	Description
<i>placeholder, x</i>	Items in italics are placeholders for information that you provide. Italicized text is also used for the titles of publications and for emphasis. Plain lowercase letters are also used as placeholders for single letters and numbers.
code	Fixed-width type indicates text that must be typed exactly as shown. It is used for instruction mnemonics, directives, symbols, subcommands, parameters, and operators. Fixed-width type is also used for example code. Instruction mnemonics and directives in text and tables are shown in all caps; for example, BSR.
SR[SCM]	A mnemonic in brackets represents a named field in a register. This example refers to the Scaling Mode (SCM) field in the Status Register (SR).
REVNO[6:4], XAD[7:0]	Numbers in brackets and separated by a colon represent either: • A subset of a register's named field For example, REVNO[6:4] refers to bits 6–4 that are part of the COREREV field that occupies bits 6–0 of the REVNO register. • A continuous range of individual signals of a bus For example, XAD[7:0] refers to signals 7–0 of the XAD bus.

1.5.3 Special terms

The following terms have special meanings:

Term	Meaning
asserted	Refers to the state of a signal as follows: • An active-high signal is asserted when high (1). • An active-low signal is asserted when low (0).
deasserted	Refers to the state of a signal as follows: • An active-high signal is deasserted when low (0). • An active-low signal is deasserted when high (1). In some cases, deasserted signals are described as <i>negated</i>.
reserved	Refers to a memory space, register, field, or programming setting. Writes to a reserved location can result in unpredictable functionality or behavior. • Do not modify the default value of a reserved programming setting, such as the reset value of a reserved register field. • Consider undefined locations in memory to be reserved.
w1c	Write 1 to clear: Refers to a register bitfield that must be written as 1 to be "cleared."

1.6 References

In addition to this reference manual, the following documents provide information about the operation of the MPC5777C:

- MPC5777C Reference Manual Addendum (document number MPC5777CRMAD)
- MPC5777C Data Sheet (document number MPC5777C)
- *e200z759n3 Core Reference Manual* (document number e200z759n3CRM)
- *Safety Manual for MPC5777C* (document number MPC5777CSM)
- IEEE-ISTO 5001-2003 Standard for a Global Embedded Processor Interface (Nexus)
- IEEE 1149.1-2001 standard - IEEE Standard Test Access Port and Boundary-Scan Architecture
- Power Architecture Book E V1.0, available at [www.nxp.com](http://www.nxp.com/files/32bit/doc/user_guide/BOOK_EUM.pdf): http://www.nxp.com/files/32bit/doc/user_guide/BOOK_EUM.pdf

Chapter 2

Introduction

2.1 Overview

The MPC5700 family of devices is closely compatible with the MPC5600 families, while introducing new features coupled with high performance CMOS technology to provide substantial cost reduction per feature and significant performance improvement. This document summarizes features and critical performance parameters of the MPC5777C.

The two e200z7 host processor cores of the MPC5777C are compatible with the Power Architecture Book E architecture. They are 100% user-mode compatible (with floating point library) with the classic PowerPC instruction set. The Book E architecture has enhancements that improve the architecture's fit in embedded applications. In addition to the standard and VLE Power Architecture instruction sets, the cores of the MPC5777C have instruction support for digital signal processing (DSP).

The MPC5777C has two levels of memory hierarchy: separate 16 KB instruction and 16 KB data caches for each of two cores and 512 KB of on-chip SRAM as well as 8 MB of internal flash memory. An external bus interface is also available for special packaged parts to support application development and calibration.

2.2 Device Description

2.2.1 Family comparison

The following table summarizes major features of the family of devices defined in this document and compares the family with predecessor families of similar devices. The feature column represents a combination of module names and capabilities of certain modules.

Table 2-1. Family comparison

Feature	MPC5565	MPC5567	MPC5566	MPC5674F	MPC5676R	MPC5777C
Process	130 nm	130 nm	130 nm	90 nm	90 nm	55 nm
Core	z6	z6	z6	z7	z7 ¹	z7 ¹
Number of cores	1	1	1	1	2	2 + 1 lockstep core
Single Precision Floating Point	Yes	Yes	Yes	Yes	Yes	Yes
SIMD	Yes	Yes	Yes	Yes	Yes	Yes
VLE	Yes	Yes	Yes	Yes	Yes	Yes
Cache	8KB Unified	8KB Unified	32KB Unified	16KB Instruction + 16KB Data	16KB Instruction + 16KB Data	16KB Instruction + 16KB Data
Non-maskable interrupt	No	No	No	NMI, Critical	NMI, Critical	NMI, Critical
MMU entries	32	32	32	64	32	32
MMU Tool Control	No	No	No	No	Yes	Yes
MPU	No	No	No	Yes	Yes	Yes
Semaphores	No	No	No	No	16	16
CRC channels	No	No	No	No	3	3
Software Watchdog Timer	No	No	No	1	2	2
Core Nexus Class	3+	3+	3+	3+	3+	3+
SRAM	64KB (32K Standby)	64KB (32K Standby)	128KB (32K Standby)	256KB (32K Standby)	384KB (48K Standby)	512KB (64K Standby)
Flash memory	2MB	2MB	3MB	4MB	6MB	8MB
Flash memory fetch accelerator	2 x 256 bit	2 x 256 bit	2 x 256 bit	4 x 256 bit	4 x 256 bit per core	16 x 256 bit per core
Tamper detection	No	No	No	No	No	Yes
Security modules	No	No	No	No	No	Yes
Safety modules	No	No	No	No	No	Yes
External bus	32 bit	32 bit	32 bit	Yes ²	Yes ²	Yes ²
Calibration bus	16 bit	16 bit	16 bit	16 bit non-mux 16, 32 bit muxed ³	16 bit non-mux 16, 32 bit muxed ³	16 bit non-mux 16, 32 bit muxed ³
DMA channels	32	32	64	64 + 32	64 + 64	64 + 64
DMA Nexus Class	3	3	3	3	3	3
Serial Interface	2	2	2	3	3	5
FlexCAN	3	5	4	4	4	4
M_CAN with FD	No	No	No	No	No	2
SPI	3	3	4	4	5	5

Table continues on the next page...

Table 2-1. Family comparison (continued)

Feature	MPC5565	MPC5567	MPC5566	MPC5674F	MPC5676R	MPC5777C
Microsecond bus downlink	No	No	No	Yes	Yes	Yes
SENT channels	No	No	No	No	No	12
PSI5 channels	No	No	No	No	No	2
Zipwire	No	No	No	No	No	1
FlexRay	No	Yes	No	Yes	Yes	No
Ethernet	No	Yes	Yes	No	No	Yes
System Timers	No	No	No	1 RTI, 4 PIT channels 4 AutoSAR	1 RTI, 4 PIT channels 4 AutoSAR	1 RTI, 4 PIT channels 4 AutoSAR
eMIOS channels	24	24	24	32	32	32
eTPU channels	32	32	64	64	96	96
eTPU version	1 x eTPU	1 x eTPU	2 x eTPU	2 x eTPU2	3 x eTPU2	3 x eTPU2
eTPU code memory	12KB	12KB	12KB	24KB	24KB + 12KB	24KB + 12KB
eTPU data memory	2.5KB	2.5KB	3KB	6KB	6KB + 3KB	6KB + 3KB
Interrupt controller	210 sources	210 sources	308 sources	448 sources	500 sources	500+ sources
ADC input pins	40	40	40	64	64	70
ADC input diagnostics	No	No	No	Yes	Yes	Yes
ADC resolution	12 bit	12 bit	12 bit	12 bit	12 bit	12 bit
ADC quantity	2	2	2	4	4	4
ADC variable gain amp.	No	No	No	Yes	Yes	Yes
Sigma-Delta ADC	No	No	No	No	No	4
Reaction channels	No	No	No	No	No	10
Temp. sensor	No	No	No	Yes	Yes	Yes
Decimation filters	No	No	No	8	12	12
Protected Port Output	No	No	No	No	4	4
Self Test Controller	No	No	No	No	Yes	Yes
Dev Tool Semaphores	No	No	No	No	32	32
PLL	FM	FM	FM	FM	FM	FM + non-FM
Integrated linear voltage regulator	1.5V	1.5V	1.5V	3.3V, 1.2V	3.3V, 1.2V	3.3V, 1.2V

Table continues on the next page...

Table 2-1. Family comparison (continued)

Feature	MPC5565	MPC5567	MPC5566	MPC5674F	MPC5676R	MPC5777C
Integrated switch mode voltage regulator	No	No	No	1.2V	1.2V	1.2V
External power supplies	5V, 3.3V	5V, 3.3V	5V, 3.3V	5V ⁴	5V ⁴	5V ⁴
Low power modes	No	No	No	Stop mode Slow mode	Stop mode Slow mode	Stop mode Slow mode

1. There are two cores on the MPC5777C and MPC5676R. Both cores have identical features.
2. External Bus Interface (EBI) is not available on 416 PBGA
3. Calibration bus is combined with EBI.
4. External 3.3V may be needed for external 3.3V pins.

2.2.2 Block diagram

The following figure shows a top-level block diagram of the MPC5777C. The purpose of the block diagram is to show the general interconnection of functional modules through the crossbar switch.

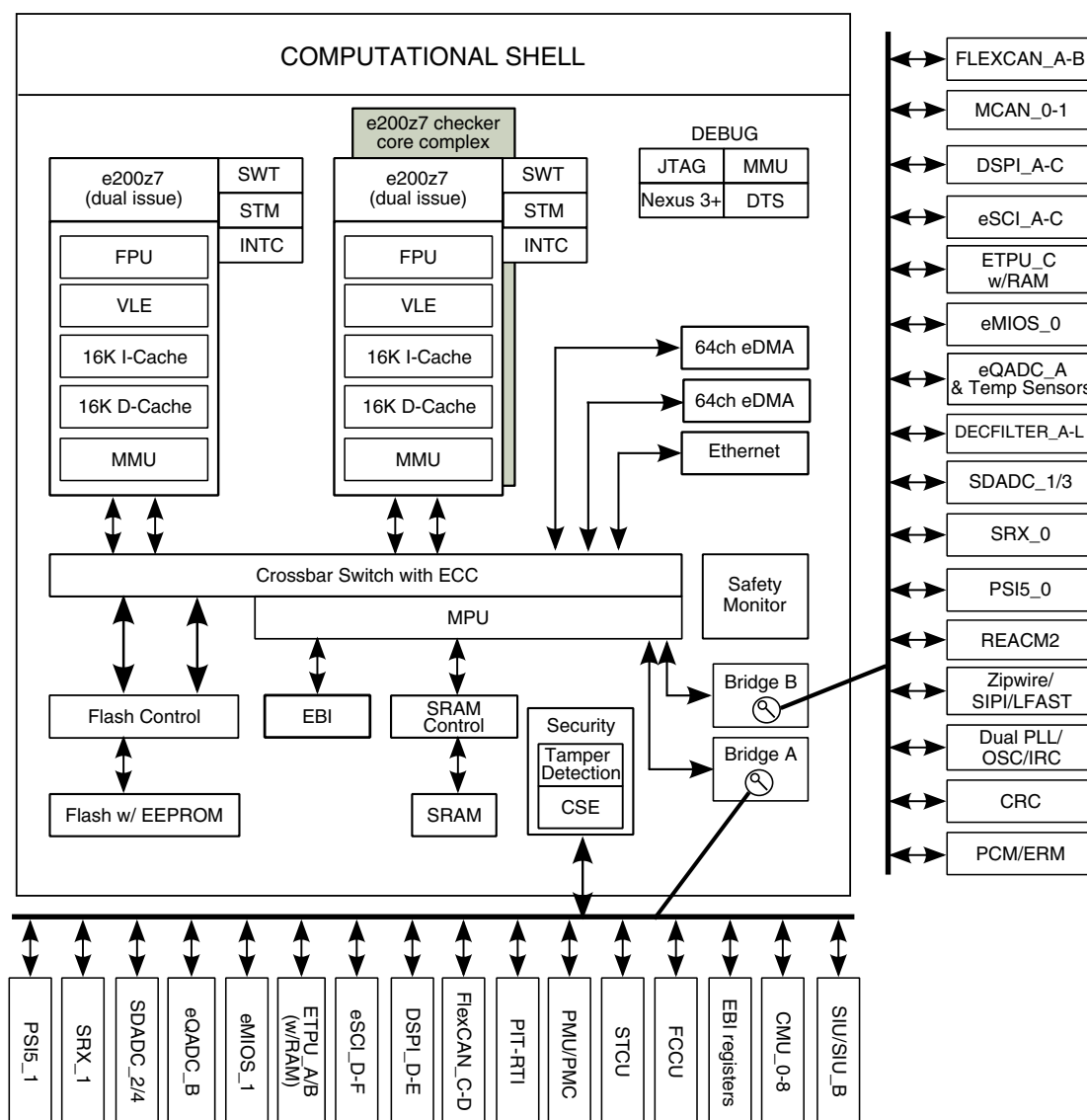


Figure 2-1. MPC5777C block diagram

2.2.3 Packages

The MPC5777C is offered in the following package types:

- 416-ball MAPBGA, 1 mm ball pitch, 27 mm × 27 mm outline, without EBI
- 516-ball MAPBGA, 1 mm ball pitch, 27 mm × 27 mm outline, with EBI
- VertiCal Calibration System, which includes EBI

2.2.4 Features summary

On-chip modules available within the family include the following features:

- Three dual issue, 32-bit CPU core complexes (e200z7), two of which run in lockstep
 - Power Architecture embedded specification compliance
 - Instruction set enhancement allowing variable length encoding (VLE), optional encoding of mixed 16-bit and 32-bit instructions, for code size footprint reduction
 - On the two computational cores: Signal processing extension (SPE1.1) instruction support for digital signal processing (DSP)
 - Single-precision floating point operations
 - On the two computational cores: 16 KB I-Cache and 16 KB D-Cache
 - Hardware cache coherency between cores
- 16 hardware semaphores
- 3-channel CRC module
- 8 MB on-chip flash memory
 - Supports read during program and erase operations, and multiple blocks allowing EEPROM emulation
- 512 KB on-chip general-purpose SRAM including 64 KB standby RAM
- Two multichannel direct memory access controllers (eDMA)
 - 64 channels per eDMA
- Dual core Interrupt Controller (INTC)
- Dual phase-locked loops (PLLs) with stable clock domain for peripherals and frequency modulation (FM) domain for computational shell
- Crossbar Switch architecture for concurrent access to peripherals, flash memory, or RAM from multiple bus masters with End-To-End ECC
- External Bus Interface (EBI) for calibration and application use
- System Integration Unit (SIU)
- Error Injection Module (EIM) and Error Reporting Module (ERM)
- Four protected port output (PPO) pins
- Boot Assist Module (BAM) supports serial bootload via CAN or SCI
- Three second-generation Enhanced Time Processor Units (eTPUs)
 - 32 channels per eTPU
 - Total of 36 KB code RAM
 - Total of 9 KB parameter RAM
- Enhanced Modular Input/Output System (eMIOS) supporting 32 unified channels with each channel capable of single action, double action, pulse width modulation (PWM) and modulus counter operation
- Two Enhanced Queued Analog-to-Digital Converter (eQADC) modules with:
 - Two separate analog converters per eQADC module
 - Support for a total of 70 analog input pins, expandable to 182 inputs with off-chip multiplexers

- Interface to twelve hardware Decimation Filters
- Enhanced "Tap" command to route any conversion to two separate Decimation Filters
- Four independent 16-bit Sigma-Delta ADCs (SDADCs)
- 10-channel Reaction Module
- Ethernet (FEC)
- Two PSI5 modules
- Two SENT Receiver (SRX) modules supporting 12 channels
- Zipwire: SIPI and LFAST modules
- Five Deserial Serial Peripheral Interface (DSPI) modules
- Five Enhanced Serial Communication Interface (eSCI) modules
- Four Controller Area Network (FlexCAN) modules
- Two M_CAN modules that support FD
- Fault Collection and Control Unit (FCCU)
- Clock Monitor Units (CMUs)
- Tamper Detection Module (TDM)
- Cryptographic Services Engine (CSE)
 - Complies with *Secure Hardware Extension (SHE) Functional Specification Version 1.1* security functions
 - Includes software selectable enhancement to key usage flag for MAC verification and increase in number of memory slots for security keys
- PASS module to support security features
- Nexus development interface (NDI) per IEEE-ISTO 5001-2003 standard, with some support for 2010 standard
- Device and board test support per Joint Test Action Group (JTAG) IEEE 1149.1 and 1149.7
- On-chip voltage regulator controller (VRC) that derives the core logic supply voltage from the high-voltage supply
- On-chip voltage regulator for flash memory
- Self Test capability

Chapter 3

Signal Description

3.1 Production packages

The following production packages are available for the MPC5777C:

- BGA416
- BGA516

The 516 BGA is a superset of the 416 BGA. Compared to the BGA416, BGA516 contains the EBI, additional GPIO and analog inputs, and added multiplexing options.

Case number and outline drawings for each package are provided in the MPC5777C data sheet.

3.2 Emulation packages

The following emulation packages are available for the MPC5777C:

- BGA552

The case outline drawing for the emulation packages is provided in the MPC5777C data sheet.

3.3 Pad ring and ball maps

Refer to the MPC5777C data sheet for ball-map figures.

3.4 Pin function selection

3.4.1 Pad output multiplexing

Most pins on the package support more than one output function. The selection of which function is driven to the pin is done by a set of Port Control Registers (SIU_PCR) within the SIU module. These registers also configure electrical attributes such as drive strength and pull up/down selection. Refer to the SIU_PCR section of the SIU chapter for more information.

3.4.2 Pin input multiplexing

Some modules in this device are allowed to receive input signals from multiple package pins. The selection of which pin drives the function is done by a set of Input Multiplexing Registers (SIU_IMUX) within the SIU module. Refer to the SIU_IMUX section of the SIU chapter for more information.

3.4.3 LVDS signal selection

A pin that is driven by a signal from a 5V pad or from an LVDS pad has a single PCR controlling both signals. Different values of PA select either the 5V signal or the LVDS signal. Note that a single LVDS pad supplies two signals: a "true" and a "complement" output. This means that one LVDS pad drives two balls, but each ball has a separate PCR (and PA field). The two LVDS signals are enabled only when the PA values in both PCRs are set to select LVDS operation. Both LVDS signals are deselected when either PA is set to a value that deselects its LVDS signal. The ball that has its PA field still set to LVDS will go to an undriven state until its PA field is updated to a non-LVDS value.

The following table is an example of the options for PA on two balls, where LVDS "true" and LVDS "complement" LVDS outputs are multiplexed with SCK_C and SINC.

Table 3-1. LVDS example

PCR235 ¹ PA selection	PCR236 ² PA selection	SCKC ball function	SINC ball function
SCKC	SINC	SCKC	SINC
LVDS	LVDS	SCK_C_LVDS_P	SCK_C_LVDS_N
SCKC	LVDS	SCKC	undriven
LVDS	SINC	undriven	SINC

1. SCKC_SCK_C_LVDS_P_GPIO235

2. SINC_SCK_C_LVDS_N_GPIO236

Table 3-2 lists all LVDS pads available on the MPC5777C and respective SIU PCR configuration registers and PA values. When LVDS TX functions are enabled on the balls, they can be configured to operate in LFAST mode or Micro-Second Bus mode. Also, LVDS RX pads can be configured to increase bias current and enhance timing performance. Table 3-3 lists the register bits used to configure LVDS pad operating modes.

NOTE

Differential signal output swing can be configured on MPC5776R LVDS pads. This feature is not available on MPC5777C LVDS pads.

Table 3-2. LVDS function selection

LVDS function	Pin polarity	Description	SIU_PCR[PA] configuration
SIPI_TX	–	SIPI Transmit LVDS Neg Terminal	PCR208[PA]=100b AND PCR209[PA]=100b
	+	SIPI Transmit LVDS Pos Terminal	
SIPI_RX	+	SIPI Receive LVDS Pos Terminal	PCR212[PA]=100b AND PCR213[PA]=100b
	–	SIPI Receive LVDS Neg Terminal	
SCK_C_LVDS	+	MSC LVDS+ downstream signal positive output clock	PCR235[PA]=10b AND PCR236[PA]=10b
	–	MSC LVDS– downstream signal negative output clock	
SOUT_C_LVDS	+	MSC LVDS+ downstream signal positive output data	PCR237[PA]=10b AND PCR238[PA]=10b
	–	MSC LVDS– downstream signal negative output data	
SCK_D_LVDS	+	MSC LVDS+ downstream signal positive output clock	PCR240[PA]=10b AND PCR241[PA]=10b
	–	MSC LVDS– downstream signal negative output clock	
SOUT_D_LVDS	+	MSC LVDS+ downstream signal positive output data	PCR242[PA]=011b AND PCR243[PA]=011b
	–	MSC LVDS– downstream signal negative output data	

Table 3-3. LVDS pad operating modes

LVDS function	Configuration register/bit	Value	Pad effect
SIPI_TX	LFAST LCR/LVTXOP	0	Micro-Second Bus mode
		1	LFAST mode
SIPI_RX	LFAST LCR/LVRXOP[0]	0	Normal Ibias current to RX comp block
		1	Increased Ibias current to RX comp block. Improves timing performance
SCK_C_LVDS	SIU PCR235/SRC[0]	0	Micro-Second Bus mode
		1	LFAST mode
SOUT_C_LVDS	SIU PCR237/SRC[0]	0	Micro-Second Bus mode
		1	LFAST mode
SCK_D_LVDS	SIU PCR240/SRC[0]	0	Micro-Second Bus mode
		1	LFAST mode

Table continues on the next page...

Table 3-3. LVDS pad operating modes (continued)

LVDS function	Configuration register/bit	Value	Pad effect
SOUT_D_LVDS	SIU PCR242/SRC[0]	0	Micro-Second Bus mode
		1	LFAST mode

3.5 External signal descriptions, multiplexing, and attributes

Details about external signal functions, their static electrical characteristics, and pad configuration settings for this chip appear in the spreadsheet that is embedded within this document:

MPC5777C_System_IO_Definition.xlsx

To access this spreadsheet, view the document's list of attachments.

The signal properties and their electrical characteristics are set in the SIU's Pad Configuration Registers (PCRs).

3.5.1 IBE and OBE control special cases

The "input buffer enable" and/or "output buffer enable" of some signals are not controlled by the IBE/OBE bits of the corresponding SIU_PCR. The following tables list these special-case signals:

- IBE always ON: [Table 3-4](#)
- OBE always ON: [Table 3-5](#)
- IBE and OBE always ON: [Table 3-6](#)
- IBE and/ or OBE controlled by a peripheral or controller: [Table 3-7](#)

Table 3-4. IBE always ON signals

Signal	Direction	PCR
BOOTCFG0	I	211
BOOTCFG1	I	212
CNRXA	I	84
CNRXB	I	86
CNRXC	I	88
CNRXD	I	247
		194
ERRORIN	I	124

Table continues on the next page...

Table 3-4. IBE always ON signals (continued)

Signal	Direction	PCR
ERROR0 ¹	I	117
ERROR1 ¹	I	125
ETRIG0	I	244
ETRIG1	I	101
IRQ0	I	450
		193
IRQ1	I	451
		194
IRQ10	I	136
IRQ11	I	137
IRQ12	I	138
IRQ13	I	139
IRQ14	I	140
IRQ15	I	141
IRQ2	I	452
		211
IRQ3	I	453
		212
IRQ4	I	454
		208
IRQ5	I	455
		209
IRQ6	I	440
		146
IRQ7	I	113
IRQ8	I	134
IRQ9	I	135
NMI	I	213
PLLCFG0 ²	I	208
PLLCFG1 ²	I	209
PLLCFG2	I	210
RXDA	I	461
		84
RXDB	I	463
		92
RXDC	I	245
RXDD	I	101
RXDE	I	88
RXDF	I	459

Table continues on the next page...

Table 3-4. IBE always ON signals (continued)

Signal	Direction	PCR
SINA	I	94
SINB	I	103
SINC	I	236
		108
SIND	I	472
		241
		190
SINE	I	99
TCRCLKA	I	113
TCRCLKB	I	146
TCRCLKC	I	445
		440
WKPCFG	I	213

1. ERROR0 and ERROR1 IBE are forced ON when DCF client is set to lock ERROR0/1 functions on respective pads. See [FCCU signals](#). When function is not locked, IBE control depends on FCCU control.
2. PLLCFG0 and PLLCFG1 have no function on MPC5777C. These pin names exist only for compatibility with MPC5676R.

Table 3-5. OBE always ON signals

Signal	Direction	PCR
CNTXA	O	83
CNTXB	O	85
CNTXC	O	87
CNTXD	O	246
		193
D_ALE	O	299
EMIOS14	O	193
EMIOS15	O	194
EMIOS4	O	183
EMIOS5	O	184
ETPUA0	O	179
ETPUA1	O	180
ETPUA12	O	114
ETPUA13	O	115
ETPUA14	O	116
ETPUA15	O	117
ETPUA16	O	118
ETPUA17	O	119
ETPUA18	O	120
ETPUA19	O	121

Table continues on the next page...

Table 3-5. OBE always ON signals (continued)

Signal	Direction	PCR
ETPUA2	O	181
ETPUA20	O	122
ETPUA21	O	123
ETPUA22	O	124
ETPUA23	O	125
ETPUA3	O	182
ETPUA4	O	183
ETPUA5	O	184
ETPUA6	O	185
ETPUA7	O	186
ETPUA8	O	187
ETPUA9	O	188
ETPUB0	O	195
ETPUB1	O	196
ETPUB16	O	147
ETPUB17	O	148
ETPUB18	O	149
ETPUB19	O	150
ETPUB2	O	197
ETPUB20	O	151
ETPUB21	O	152
ETPUB22	O	153
ETPUB23	O	154
ETPUB24	O	155
ETPUB25	O	156
ETPUB26	O	157
ETPUB27	O	158
ETPUB28	O	159
ETPUB29	O	160
ETPUB3	O	198
ETPUB30	O	161
ETPUB31	O	162
ETPUB4	O	199
ETPUB5	O	200
ETPUB6	O	201
ETPUB7	O	202
MAA0	O	464
MAA1	O	465
MAA2	O	466

Table continues on the next page...

Table 3-5. OBE always ON signals (continued)

Signal	Direction	PCR
MAB0	O	467
		464
MAB1	O	468
		465
MAB2	O	469
		466
MCKO	O	219
MDO0	O	220
PCSA1	O	163
		97
PCSA2	O	164
		98
PCSA3	O	165
		99
PCSA4	O	166
		100
PCSA5	O	101
PCSB1	O	204
		126
		106
PCSB2	O	432
		107
PCSB3	O	433
		127
		108
PCSB4	O	128
		109
PCSB5	O	129
		110
PCSC1	O	435
		239
		142
		93
PCSC2	O	436
		240
		143
		94

Table continues on the next page...

Table 3-5. OBE always ON signals (continued)

Signal	Direction	PCR
PCSC3	O	241
		144
		85
PCSC4	O	242
		145
		86
PCSC5	O	437
		243
		95
PCSD1	O	468
		130
		91
PCSD2	O	467
		131
		105
		96
PCSD3	O	466
		132
		87
PCSD4	O	465
		133
		88
PCSD5	O	464
		92
PCSE1	O	210
PCSE2	O	446
		180
PCSE3	O	447
		181
PCSE4	O	448
		182
PCSE5	O	449
		183
RDY	O	226
RSTOUT	O	230
SCK_C_LVDS+	O	235
SCK_D_LVDS+	O	240
SOUT_C_LVDS+	O	237
SOUT_D_LVDS+	O	242

Table continues on the next page...

Table 3-5. OBE always ON signals (continued)

Signal	Direction	PCR
TXDD	O	99

Table 3-6. IBE and OBE always ON signals

Signal	Direction	PCR
TXDA	I/O	460
		89
		83
TXDB	I/O	462
		91
TXDC	I/O	244
TXDD	I/O	239
TXDE	I/O	223
		100
TXDE	I/O	223
TXDF	I/O	458

Table 3-7. IBE and/or OBE controlled by peripheral

Controlled by	Signal	Direction	IBE/OBE
LFAST/SIPI	SIPI_RX	I	IBE
LFAST/SIPI	SIPI_TX	O	OBE
DSPI_A	SOUTA	O	OBE
DSPI_B	SOUTB	O	OBE
DSPI_C	SOUTC	O	OBE
DSPI_D	SOUTD	O	OBE
DSPI_E	SOUTE	O	OBE
EBI	D_ADD_DAT16	I/O	IBE and OBE
EBI	D_ADD_DAT17	I/O	IBE and OBE
EBI	D_ADD_DAT18	I/O	IBE and OBE
EBI	D_ADD_DAT19	I/O	IBE and OBE
EBI	D_ADD_DAT20	I/O	IBE and OBE
EBI	D_ADD_DAT21	I/O	IBE and OBE
EBI	D_ADD_DAT22	I/O	IBE and OBE
EBI	D_ADD_DAT23	I/O	IBE and OBE
EBI	D_ADD_DAT24	I/O	IBE and OBE
EBI	D_ADD_DAT25	I/O	IBE and OBE
EBI	D_ADD_DAT26	I/O	IBE and OBE
EBI	D_ADD_DAT27	I/O	IBE and OBE

Table continues on the next page...

Table 3-7. IBE and/or OBE controlled by peripheral (continued)

Controlled by	Signal	Direction	IBE/OBE
EBI	D_ADD_DAT28	I/O	IBE and OBE
EBI	D_ADD_DAT29	I/O	IBE and OBE
EBI	D_ADD_DAT30	I/O	IBE and OBE
EBI	D_ADD_DAT31	I/O	IBE and OBE
EBI	D_ADD10	O	OBE
EBI	D_ADD11	O	OBE
EBI	D_ADD12	O	OBE
EBI	D_ADD13	O	OBE
EBI	D_ADD14	O	OBE
EBI	D_ADD15	O	OBE
EBI	D_ADD16	O	OBE
EBI	D_ADD17	O	OBE
EBI	D_ADD18	O	OBE
EBI	D_ADD19	O	OBE
EBI	D_ADD20	O	OBE
EBI	D_ADD21	O	OBE
EBI	D_ADD22	O	OBE
EBI	D_ADD23	O	OBE
EBI	D_ADD24	O	OBE
EBI	D_ADD25	O	OBE
EBI	D_ADD26	O	OBE
EBI	D_ADD27	O	OBE
EBI	D_ADD28	O	OBE
EBI	D_ADD29	O	OBE
EBI	D_ADD30	O	OBE
EBI	D_ADD9	O	OBE
EBI	D_BDIP	O	OBE
EBI	D_CS0	O	OBE
EBI	D_CS1	O	OBE
EBI	D_CS2	O	OBE
EBI	D_CS3	O	OBE
EBI	D_DAT0	I/O	IBE and OBE
EBI	D_DAT1	I/O	IBE and OBE
EBI	D_DAT10	I/O	IBE and OBE
EBI	D_DAT11	I/O	IBE and OBE
EBI	D_DAT12	I/O	IBE and OBE
EBI	D_DAT13	I/O	IBE and OBE
EBI	D_DAT14	I/O	IBE and OBE
EBI	D_DAT15	I/O	IBE and OBE

Table continues on the next page...

Table 3-7. IBE and/or OBE controlled by peripheral (continued)

Controlled by	Signal	Direction	IBE/OBE
EBI	D_DAT2	I/O	IBE and OBE
EBI	D_DAT3	I/O	IBE and OBE
EBI	D_DAT4	I/O	IBE and OBE
EBI	D_DAT5	I/O	IBE and OBE
EBI	D_DAT6	I/O	IBE and OBE
EBI	D_DAT7	I/O	IBE and OBE
EBI	D_DAT8	I/O	IBE and OBE
EBI	D_DAT9	I/O	IBE and OBE
EBI	D_OE	O	OBE
EBI	D_RD_WR	O	OBE
EBI	D_TA	I/O	IBE and OBE
EBI	D_TEA	I/O	IBE and OBE
EBI	D_TS	O	OBE
EBI	D_WE0	O	OBE
EBI	D_WE1	O	OBE
EBI	D_WE2	O	OBE
EBI	D_WE3	O	OBE
Fast Ethernet Controller	FEC_MDIO	I/O	OBE
FCCU	ERROR0	I/O	IBE and OBE
FCCU	ERROR1	I/O	IBE and OBE
JTAG Controller	TDO	O	OBE
JTAG Controller	TMS	I	OBE
Nexus Controller	EVTO	O	OBE
Nexus Controller	MDO1	O	OBE
Nexus Controller	MDO10	O	OBE
Nexus Controller	MDO11	O	OBE
Nexus Controller	MDO12	O	OBE
Nexus Controller	MDO13	O	OBE
Nexus Controller	MDO14	O	OBE
Nexus Controller	MDO15	O	OBE
Nexus Controller	MDO2	O	OBE
Nexus Controller	MDO3	O	OBE
Nexus Controller	MDO4	O	OBE
Nexus Controller	MDO5	O	OBE
Nexus Controller	MDO6	O	OBE
Nexus Controller	MDO7	O	OBE
Nexus Controller	MDO8	O	OBE
Nexus Controller	MDO9	O	OBE
Nexus Controller	MSEO0	O	OBE

Table continues on the next page...

Table 3-7. IBE and/or OBE controlled by peripheral (continued)

Controlled by	Signal	Direction	IBE/OBE
Nexus Controller	MSEO1	O	OBE

3.5.2 FCCU signals

A DCF client, as defined in the DCF Records chapter, is used to enable ERROR0, ERROR1, and ERRORIN functions on each pin independently. If one of the ERROR functions is disabled by the DCF client at power-on reset (POR), the associated pin resets as GPIO.

If the ERROR0, ERROR1, or ERRORIN function is enabled on the pin by the DCF client bits E0E, E1E, or EINE, respectively, the ERROR function and its attributes are locked on the pin until the next POR. A reset of the device that is not a POR will not disrupt the ERROR pins' function or state.

When the ERROR function is enabled by the DCF client at POR, the following applies to the ERROR0 and ERROR1 pins:

- POR state of the pins is Hi-Z
- Input to the FCCU is enabled
- FCCU drives the output data on the pin and no muxing change can occur on the pin, but software can read the pin state via GPIO if it enables the input buffer in the corresponding PCR
- Bits in the DCF client select the open-drain or push-pull output configuration (see the DCF Records chapter for details):
 - E1OD (ERROR1)
 - E0OD (ERROR0)
- The slew rate of the ERROR0/1 pins is set to the maximum
- Weak pull up/down is disabled, which overrides the weak pull up/down configuration for the GPIO pins determined by the WKPCFG pin or WKPCFG DCF client

When the ERROR function is enabled by the DCF client at POR, the following applies to the ERRORIN pin:

- POR state of the pins is Hi-Z
- Input to the FCCU is enabled
- Weak pull up/down is disabled, which overrides the weak pull up/down configuration for the GPIO pins determined by the WKPCFG pin or WKPCFG DCF client

Chapter 4

Memory Map, Interrupts, and DMA Requests

4.1 Memory map

All addresses in the device are identified in the following table. The addresses represent the physical addresses assigned to each block. Logical addresses are translated by the MMU into physical addresses.

Under software control of the Memory Management Unit (MMU), the logical addresses allocated to IP blocks may be changed on a minimum of a 1 KB boundary.

NOTE

Address ranges not declared in the memory map table must be considered reserved.

Table 4-1. Memory map

Address range	Module
Flash memory—see Table 4-2 for details	
0x0000_0000–0x0003_FFFF	Flash memory Low/Mid Blocks (256K)
0x0004_0000–0x003F_FFFF	Reserved
0x0040_0000–0x0040_3FFF	Flash memory UTEST (16K)—see Table 4-3 for details
0x0040_4000–0x005F_FFFF	Reserved
0x0060_0000–0x0060_7FFF	Flash memory High Blocks (32K)
0x0060_8000–0x007F_FFFF	Reserved
0x0080_0000–0x00FF_FFFF	Flash memory Large Blocks (8M)
0x0100_0000–0x0FFF_FFFF	Reserved
0x1000_0000–0x1FFF_FFFF	Emulation remapping of flash memory using MMU
External memory	
0x2000_0000–0x3FFF_FFFF	External Bus Memory
System RAM	
0x4000_0000–0x4000_FFFF	Standby SRAM on XBAR Slave Port 2 (64 KB)
0x4001_0000–0x4003_FFFF	SRAM on XBAR Slave Port 2 (192 KB)

Table continues on the next page...

Table 4-1. Memory map (continued)

Address range	Module
0x4004_0000–0x4007_FFFF	SRAM on XBAR Slave Port 4 (256 KB)
0x4008_0000–0xC3DF_FFFF	Reserved
Peripheral Bridge A (AIPS_1)	
0xC3E0_0000–0xC3E0_3FFF	Reserved
0xC3E0_4000–0xC3E0_7FFF	Input Glitch Filter 1
0xC3E0_8000–0xC3E1_3FFF	Reserved
0xC3E1_4000–0xC3E1_7FFF	PSI5_1
0xC3E1_8000–0xC3E4_3FFF	Reserved
0xC3E4_4000–0xC3E4_7FFF	SDADC_2
0xC3E4_8000–0xC3E4_BFFF	SDADC_4
0xC3E4_C000–0xC3E5_3FFF	Reserved
0xC3E5_4000–0xC3E5_7FFF	eQADC_B
0xC3E5_8000–0xC3E5_BFFF	Reserved
0xC3E5_C000–0xC3E5_FFFF	SRX_1
0xC3E6_0000–0xC3E6_3FFF	FlexCAN_C
0xC3E6_4000–0xC3E6_7FFF	FlexCAN_D
0xC3E6_8000–0xC3E6_C1FF	Reserved
0xC3E6_C200–0xC3E6_C23F	CMU_0
0xC3E6_C240–0xC3E6_C27F	CMU_1
0xC3E6_C280–0xC3E6_C2BF	CMU_2
0xC3E6_C2C0–0xC3E6_C2FF	CMU_3
0xC3E6_C300–0xC3E6_C33F	CMU_4
0xC3E6_C340–0xC3E6_C37F	CMU_5
0xC3E6_C380–0xC3E6_C3BF	CMU_6
0xC3E6_C3C0–0xC3E6_C3FF	CMU_7
0xC3E6_C400–0xC3E6_C43F	CMU_8
0xC3E7_C440–0xC3EF_FFFF	Not allocated
0xC3F0_0000–0xC3F0_3FFF	PBRIDGE registers
0xC3F0_4000–0xC3F7_FFFF	Reserved ¹
0xC3F8_0000–0xC3F8_3FFF	FCCU
0xC3F8_4000–0xC3F8_7FFF	EBI Configuration
0xC3F8_8000–0xC3F8_FFFF	Reserved
0xC3F9_0000–0xC3F9_3FFF	SIU
0xC3F9_4000–0xC3F9_7FFF	Reserved
0xC3F9_8000–0xC3F9_BFFF	SIU_B
0xC3F9_C000–0xC3F9_FFFF	Reserved
0xC3FA_0000–0xC3FA_3FFF	eMIOS_1
0xC3FA_4000–0xC3FA_FFFF	Reserved
0xC3FB_0000–0xC3FB_3FFF	eSCI_D

Table continues on the next page...

Table 4-1. Memory map (continued)

Address range	Module
0xC3FB_4000–0xC3FB_7FFF	eSCI_E
0xC3FB_8000–0xC3FB_BFFF	eSCI_F
0xC3FB_C000–0xC3FB_FFFF	PMC
0xC3FC_0000–0xC3FC_3FFF	eTPU_A/B
0xC3FC_4000–0xC3FC_7FFF	Reserved
0xC3FC_8000–0xC3FC_BFFF	eTPU Parameter RAM
0xC3FC_C000–0xC3FC_FFFF	eTPU Parameter RAM Mirror
0xC3FD_0000–0xC3FD_3FFF	eTPU Code RAM
0xC3FD_4000–0xC3FD_7FFF	eTPU Code RAM
0xC3FD_8000–0xC3FD_BFFF	Reserved
0xC3FD_C000–0xC3FD_FFFF	DSPI_D
0xC3FE_0000–0xC3FE_3FFF	DSPI_E
0xC3FE_4000–0xC3FE_FFFF	Reserved
0xC3FF_0000–0xC3FF_3FFF	PIT_RTI
0xC3FF_4000–0xC3FF_7FFF	STCU
0xC3FF_8000–0xFFDF_FFFF	Reserved
Peripheral Bridge B (AIPS_0)	
0xFFE0_0000–0xFFE0_3FFF	eTPU_C
0xFFE0_4000–0xFFE0_7FFF	REACM2
0xFFE0_8000–0xFFE0_BFFF	eTPU_C Parameter RAM
0xFFE0_C000–0xFFE0_FFFF	eTPU_C Parameter RAM Mirror
0xFFE1_0000–0xFFE1_3FFF	eTPU_C Code RAM
0xFFE1_4000–0xFFE1_FFFF	Reserved
0xFFE2_0000–0xFFE2_3FFF	Input Glitch Filter 0
0xFFE2_4000–0xFFE2_FFFF	Reserved
0xFFE3_0000–0xFFE3_3FFF	M_CAN_0
0xFFE3_4000–0xFFE3_7FFF	M_CAN RAM
0xFFE3_8000–0xFFE3_BFFF	M_CAN_1
0xFFE3_C000–0xFFE3_FFFF	PSI5_0
0xFFE4_0000–0xFFE4_3FFF	Reserved
0xFFE4_4000–0xFFE4_7FFF	SDADC_1
0xFFE4_8000–0xFFE4_BFFF	SDADC_3
0xFFE4_C000–0xFFE5_BFFF	Reserved
0xFFE5_C000–0xFFE5_FFFF	SRX_0
0xFFE6_0000–0xFFE6_3FFF	Reserved
0xFFE6_4000–0xFFE6_7FFF	eMIOS_0
0xFFE6_8000–0xFFE6_BFFF	CRC
0xFFE6_C000–0xFFE6_FFFF	SIPI
0xFFE7_0000–0xFFE7_3FFF	PCS

Table continues on the next page...

Table 4-1. Memory map (continued)

Address range	Module
0xFFE7_4000–0xFFE7_7FFF	PLLDIG
0xFFE7_8000–0xFFE7_BFFF	DTS
0xFFE7_C000–0xFFE7_FFFF	JDC
0xFFE8_0000–0xFFEF_FFFF	Not allocated ²
0xFFFF0_0000–0xFFFF0_3FFF	PBRIDGE registers
0xFFFF0_4000–0xFFFF0_7FFF	XBAR
0xFFFF0_8000–0xFFFF0_BFFF	XBIC
0xFFFF0_C000–0xFFFF0_FFFF	MPU_0
0xFFFF1_0000–0xFFFF1_3FFF	MPU_1
0xFFFF1_4000–0xFFFF1_7FFF	CSE
0xFFFF1_8000–0xFFFF1_BFFF	TDM
0xFFFF1_C000–0xFFFF1_FFFF	PRAMC_0
0xFFFF2_0000–0xFFFF2_3FFF	PRAMC_1
0xFFFF2_4000–0xFFFF2_7FFF	SEMA4
0xFFFF2_8000–0xFFFF3_3FFF	Reserved ¹
0xFFFF3_4000–0xFFFF3_7FFF	SWT_B
0xFFFF3_8000–0xFFFF3_BFFF	SWT_A
0xFFFF3_C000–0xFFFF3_FFFF	STM_A
0xFFFF4_0000–0xFFFF4_3FFF	Reserved ¹
0xFFFF4_4000–0xFFFF4_7FFF	eDMA_A
0xFFFF4_8000–0xFFFF4_BFFF	INTC
0xFFFF4_C000–0xFFFF4_FFFF	FEC
0xFFFF5_0000–0xFFFF5_3FFF	PCU
0xFFFF5_4000–0xFFFF5_7FFF	eDMA_B
0xFFFF5_8000–0xFFFF5_FFFF	Reserved ¹
0xFFFF6_0000–0xFFFF6_3FFF	ERM
0xFFFF6_4000–0xFFFF6_7FFF	EIM
0xFFFF6_8000–0xFFFF6_BFFF	PFLASH Configuration
0xFFFF6_C000–0xFFFF6_C1FF	PCM
0xFFFF6_C200–0xFFFF7_FFFF	Reserved ¹
0xFFFF8_0000–0xFFFF8_3FFF	eQADC_A
0xFFFF8_4000–0xFFFF8_7FFF	Reserved
0xFFFF8_8000–0xFFFF8_87FF	Decimation Filter A
0xFFFF8_8800–0xFFFF8_8FFF	Decimation Filter B
0xFFFF8_9000–0xFFFF8_97FF	Decimation Filter C
0xFFFF8_9800–0xFFFF8_9FFF	Decimation Filter D
0xFFFF8_A000–0xFFFF8_A7FF	Decimation Filter E
0xFFFF8_A800–0xFFFF8_AFFF	Decimation Filter F
0xFFFF8_B000–0xFFFF8_B7FF	Decimation Filter G

Table continues on the next page...

Table 4-1. Memory map (continued)

Address range	Module
0xFFFF8_B800–0xFFFF8_BFFF	Decimation Filter H
0xFFFF8_C000–0xFFFF8_C7FF	Decimation Filter I
0xFFFF8_C800–0xFFFF8_CFFF	Decimation Filter J
0xFFFF8_D000–0xFFFF8_D7FF	Decimation Filter K
0xFFFF8_D800–0xFFFF8_DFFF	Decimation Filter L
0xFFFF8_E000–0xFFFF8_FFFF	Not used
0xFFFF9_0000–0xFFFF9_3FFF	DSPI_A
0xFFFF9_4000–0xFFFF9_7FFF	DSPI_B
0xFFFF9_8000–0xFFFF9_BFFF	DSPI_C
0xFFFF9_C000–0xFFFFA_FFFF	Reserved
0xFFFFB_0000–0xFFFFB_3FFF	eSCI_A
0xFFFFB_4000–0xFFFFB_7FFF	eSCI_B
0xFFFFB_8000–0xFFFFB_BFFF	eSCI_C
0xFFFFB_C000–0xFFFFB_FFFF	Reserved
0xFFFFC_0000–0xFFFFC_3FFF	FlexCAN_A
0xFFFFC_4000–0xFFFFC_7FFF	FlexCAN_B
0xFFFFC_8000–0xFFFFD_7FFF	Reserved
0xFFFFD_8000–0xFFFFD_BFFF	LFAST
0xFFFFD_C000–0xFFFFE_3FFF	Reserved
0xFFFFE_4000–0xFFFFE_7FFF	Reserved
0xFFFFE_8000–0xFFFFE_BFFF	C55FMC flash memory module
0xFFFFE_C000–0xFFFFE_FFFF	Reserved
0xFFFFF_0000–0xFFFFF_3FFF	ATX
0xFFFFF_4000–0xFFFFF_7FFF	PASS
0xFFFFF_8000–0xFFFFF_BFFF	SSCM
0xFFFFF_C000–0xFFFFF_FFFF	BAM

- Any attempted access to this reserved space, which is part of the memory map for on-platform peripherals, generates a transfer error.
- Any attempted access to this unallocated space generates a transfer error.

Table 4-2. Flash memory map

Start address	End address	Allocated size (KB)	RWW partition	Block number	Block type	Example use
0x00000000	0x0000FFFF	64	0	0	Low	EEPROM data
0x00010000	0x0001FFFF	64	1	1	Low	EEPROM data
0x00020000	0x0002FFFF	64	2	0	Mid	EEPROM data
0x00030000	0x0003FFFF	64	3	1	Mid	EEPROM data
0x00040000	0x003FFFFF	3840	—	—	Not available	n/a
0x00400000	0x00403FFF	16	0	0	UTEST	See Table 4-3

Table continues on the next page...

Table 4-2. Flash memory map (continued)

Start address	End address	Allocated size (KB)	RWW partition	Block number	Block type	Example use
0x00404000	0x005FFFFFFF	2032	—	—	Not available	n/a
0x00600000	0x00603FFF	16	4	0	High	CSE
0x00604000	0x00607FFF	16	5	1	High	CSE
0x00608000	0x007FFFFFFF	2016	—	—	Not available	n/a
0x00800000	0x0083FFFF	256	6	0	Large	Boot
0x00840000	0x0087FFFF	256	6	1	Large	Calibration
0x00880000	0x008BFFFF	256	6	2	Large	Calibration
0x008C0000	0x008FFFFFFF	256	6	3	Large	Calibration
0x00900000	0x0093FFFF	256	6	4	Large	Calibration
0x00940000	0x0097FFFF	256	6	5	Large	Application code
0x00980000	0x009BFFFF	256	6	6	Large	Application code
0x009C0000	0x009FFFFFFF	256	6	7	Large	Application code
0x00A00000	0x00A3FFFF	256	7	8	Large	Application code
0x00A40000	0x00A7FFFF	256	7	9	Large	Application code
0x00A80000	0x00ABFFFF	256	7	10	Large	Application code
0x00AC0000	0x00AFFFFFFF	256	7	11	Large	Application code
0x00B00000	0x00B3FFFF	256	7	12	Large	Application code
0x00B40000	0x00B7FFFF	256	7	13	Large	Application code
0x00B80000	0x00BBFFFF	256	7	14	Large	Application code
0x00BC0000	0x00BFFFFFFF	256	7	15	Large	Application code
0x00C00000	0x00C3FFFF	256	8	16	Large	Application code
0x00C40000	0x00C7FFFF	256	8	17	Large	Application code
0x00C80000	0x00CBFFFF	256	8	18	Large	Application code
0x00CC0000	0x00CFFFFFFF	256	8	19	Large	Application code
0x00D00000	0x00D3FFFF	256	8	20	Large	Application code
0x00D40000	0x00D7FFFF	256	8	21	Large	Application code
0x00D80000	0x00DBFFFF	256	8	22	Large	Application code
0x00DC0000	0x00DFFFFFFF	256	8	23	Large	Application code
0x00E00000	0x00E3FFFF	256	9	24	Large	Application code
0x00E40000	0x00E7FFFF	256	9	25	Large	Application code
0x00E80000	0x00EBFFFF	256	9	26	Large	Application code
0x00EC0000	0x00EFFFFFFF	256	9	27	Large	Application code
0x00F00000	0x00F3FFFF	256	9	28	Large	Application code
0x00F40000	0x00F7FFFF	256	9	29	Large	Application code
0x00F80000	0x00FBFFFF	256	9	30	Large	Application code
0x00FC0000	0x00FFFFFFF	256	9	31	Large	Application code

Table 4-3. UTEST flash memory map

Start address	End address	Size (bytes)	Description	Notes
0x00400000	0x00400001	2	Temp Sensor 0 Calibration A	1
0x00400002	0x00400003	2	Temp Sensor 0 Calibration B	1
0x00400004	0x00400005	2	Temp Sensor 1 Calibration A	1
0x00400006	0x00400007	2	Temp Sensor 1 Calibration B	1
0x00400008	0x0040000B	4	Reserved	—
0x0040000C	0x0040000F	4	Test Mode Disable Seal	2
0x00400010	0x0040001F	16	Test Mode Disable Block Select Group A	3
0x00400020	0x0040002F	16	Reserved	—
0x00400030	0x0040003F	16	Test Mode Disable Block Select Group B	4
0x00400040	0x0040005F	32	Customer Single Bit Correction Area	5
0x00400060	0x0040007F	32	Customer Double Bit Detection Area	5
0x00400080	0x0040009F	32	Customer EDC after ECC Area	5
0x004000A0	0x004000BF	32	Unique Chip Identifier (UID)	6
0x004000C0	0x004000C3	4	Reserved	—
0x004000C4	0x004000C7	4	Reserved	—
0x004000C8	0x004000FF	56	Reserved	—
0x00400100	0x00400103	4	Test Mode Disable Override Passcode	7
0x00400104	0x0040011F	28	Reserved	7
0x00400120	0x0040013F	32	JTAG Password	7
0x00400140	0x0040015F	32	PASS Password Group 0	7
0x00400160	0x0040017F	32	PASS Password Group 1	7
0x00400180	0x0040019F	32	PASS Password Group 2	7
0x004001A0	0x004001BF	32	PASS Password Group 3	7
0x004001C0	0x004001DF	32	Reserved	7
0x004001E0	0x004001FF	32	Reserved	7
0x00400200	0x0040020F	16	Life cycle slot 0 - MCU Production	8
0x00400210	0x0040021F	16	Life cycle slot 1 - Customer Delivery	9
0x00400220	0x0040022F	16	Life cycle slot 2 - OEM Production	10
0x00400230	0x0040023F	16	Life cycle slot 3 - In Field	11
0x00400240	0x0040024F	16	Life cycle slot 4 - Failure Analysis	12
0x00400250	0x004002FF	176	Reserved	—
0x00400300	0x00400307	8	DCF start record	13
0x00400308	0x00400FFF	3320	DCF records	14
0x00401000	0x00403FFF	12288	Reserved for customer OTP data	15

1. Programmed by NXP during test. Contains the temperature sensor and bandgap ADC readings to be used by software to improve the accuracy of the temp sensor readings.
2. Programmed by NXP during test. In production devices, UTEST is sealed (one-time programmable) from the factory.
3. Programmed by the customer to select which blocks cannot be tested in Failure Analysis.
4. Secondary location that can be programmed by the customer to select which blocks cannot be tested in Failure Analysis.
5. Programmed by NXP to include ECC/EDC errors to allow testing of ECC/EDC hardware.

Interrupt vector assignments

6. Programmed by NXP during test. The UID includes information on wafer lot, X/Y-position on the wafer, manufacturing data, test results, and company ID (65 for NXP).
7. Protected from read operations by flash memory BIU from OEM Production life cycle. Programmed by customer. Becomes unreadable (hidden) as soon as customer programs life cycle to OEM Production or later.
8. MCU Production life cycle slot. Programmed by NXP during test.
9. Customer Delivery life cycle slot. Programmed by NXP during test.
10. OEM Production life cycle slot. Programmed by customer during ECO module production/test.
11. In Field life cycle slot. Programmed by customer during ECO module production/test.
12. Failure Analysis life cycle slot. Programmed by the customer on returned modules.
13. DCF start record. Programmed by NXP during test. Contiguous list of DCF records start at this address.
14. Contiguous list of DCF records. Initial records programmed by NXP. Subsequent records added to the end of the list by the customer.
15. Programmed by the customer.

4.2 Interrupt vector assignments

NOTE

The following table shows the mapping between interrupt sources and interrupt vectors. Some interrupt sources have a separate mechanism for enabling the interrupt, such as an interrupt enable register field. To support the mapping between a source and a vector, any such separate mechanism must be set to enable the interrupt source.

Table 4-4. Interrupt vectors and request sources

Offset	Vector	Source	Module	Description
0x0000	0	INTC_SSCIR0[CLR0]	Software	INTC software settable Clear flag 0
0x0010	1	INTC_SSCIR1[CLR1]	Software	INTC software settable Clear flag 1
0x0020	2	INTC_SSCIR2[CLR2]	Software	INTC software settable Clear flag 2
0x0030	3	INTC_SSCIR3[CLR3]	Software	INTC software settable Clear flag 3
0x0040	4	INTC_SSCIR4[CLR4]	Software	INTC software settable Clear flag 4
0x0050	5	INTC_SSCIR5[CLR5]	Software	INTC software settable Clear flag 5
0x0060	6	INTC_SSCIR6[CLR6]	Software	INTC software settable Clear flag 6
0x0070	7	INTC_SSCIR7[CLR7]	Software	INTC software settable Clear flag 7
0x0080	8	SWTA_IR[TIF]	Watchdog A	Software Watchdog A Interrupt flag
0x0090	9	FCCU_IRQ_STAT[ALRM_STAT] FCCU_IRQ_STAT[CFG_TO_STAT]	FCCU	FCCU ALARM state entry FCCU CONFIG state watchdog timeout
0x00A0	10	EDMA_ERL[ERR31:ERR0]	eDMA_A	eDMA channel Error flags 0-31
0x00B0	11	EDMA_IRQRL[INT00]	eDMA_A	eDMA channel Interrupt 0
0x00C0	12	EDMA_IRQRL[INT01]	eDMA_A	eDMA channel Interrupt 1
0x00D0	13	EDMA_IRQRL[INT02]	eDMA_A	eDMA channel Interrupt 2
0x00E0	14	EDMA_IRQRL[INT03]	eDMA_A	eDMA channel Interrupt 3
0x00F0	15	EDMA_IRQRL[INT04]	eDMA_A	eDMA channel Interrupt 4
0x0100	16	EDMA_IRQRL[INT05]	eDMA_A	eDMA channel Interrupt 5

Table continues on the next page...

Table 4-4. Interrupt vectors and request sources (continued)

Offset	Vector	Source	Module	Description
0x0110	17	EDMA_IRQRL[INT06]	eDMA_A	eDMA channel Interrupt 6
0x0120	18	EDMA_IRQRL[INT07]	eDMA_A	eDMA channel Interrupt 7
0x0130	19	EDMA_IRQRL[INT08]	eDMA_A	eDMA channel Interrupt 8
0x0140	20	EDMA_IRQRL[INT09]	eDMA_A	eDMA channel Interrupt 9
0x0150	21	EDMA_IRQRL[INT10]	eDMA_A	eDMA channel Interrupt 10
0x0160	22	EDMA_IRQRL[INT11]	eDMA_A	eDMA channel Interrupt 11
0x0170	23	EDMA_IRQRL[INT12]	eDMA_A	eDMA channel Interrupt 12
0x0180	24	EDMA_IRQRL[INT13]	eDMA_A	eDMA channel Interrupt 13
0x0190	25	EDMA_IRQRL[INT14]	eDMA_A	eDMA channel Interrupt 14
0x01A0	26	EDMA_IRQRL[INT15]	eDMA_A	eDMA channel Interrupt 15
0x01B0	27	EDMA_IRQRL[INT16]	eDMA_A	eDMA channel Interrupt 16
0x01C0	28	EDMA_IRQRL[INT17]	eDMA_A	eDMA channel Interrupt 17
0x01D0	29	EDMA_IRQRL[INT18]	eDMA_A	eDMA channel Interrupt 18
0x01E0	30	EDMA_IRQRL[INT19]	eDMA_A	eDMA channel Interrupt 19
0x01F0	31	EDMA_IRQRL[INT20]	eDMA_A	eDMA channel Interrupt 20
0x0200	32	EDMA_IRQRL[INT21]	eDMA_A	eDMA channel Interrupt 21
0x0210	33	EDMA_IRQRL[INT22]	eDMA_A	eDMA channel Interrupt 22
0x0220	34	EDMA_IRQRL[INT23]	eDMA_A	eDMA channel Interrupt 23
0x0230	35	EDMA_IRQRL[INT24]	eDMA_A	eDMA channel Interrupt 24
0x0240	36	EDMA_IRQRL[INT25]	eDMA_A	eDMA channel Interrupt 25
0x0250	37	EDMA_IRQRL[INT26]	eDMA_A	eDMA channel Interrupt 26
0x0260	38	EDMA_IRQRL[INT27]	eDMA_A	eDMA channel Interrupt 27
0x0270	39	EDMA_IRQRL[INT28]	eDMA_A	eDMA channel Interrupt 28
0x0280	40	EDMA_IRQRL[INT29]	eDMA_A	eDMA channel Interrupt 29
0x0290	41	EDMA_IRQRL[INT30]	eDMA_A	eDMA channel Interrupt 30
0x02A0	42	EDMA_IRQRL[INT31]	eDMA_A	eDMA channel Interrupt 31
0x02B0	43	SIU_PCSIFR[PCSI]	PCS	Progressive Clock Switch Interrupt
0x02C0	44	PLL0_SR[LOLF] PLL1_SR[LOLF]	PLLDIG	PLL Loss of Lock Flags
0x02D0	45	SIU_OSR[OVF15:OVF0]	SIU	SIU combined overrun interrupt requests of the external interrupt Overrun Flags
0x02E0	46	SIU_EIISR(EIF0)	SIU	SIU External Interrupt Flag 0
0x02F0	47	SIU_EIISR(EIF1)	SIU	SIU External Interrupt Flag 1
0x0300	48	SIU_EIISR(EIF2)	SIU	SIU External Interrupt Flag 2
0x0310	49	SIU_EIISR(EIF3)	SIU	SIU External Interrupt Flag 3
0x0320	50	SIU_EIISR(EIF15:EIF4)	SIU	SIU External Interrupt Flags 15-4
0x0330	51	eMIOS_0_GFLAG[F0]	eMIOS_0	eMIOS_0 channel 0 Flag
0x0340	52	eMIOS_0_GFLAG[F1]	eMIOS_0	eMIOS_0 channel 1 Flag
0x0350	53	eMIOS_0_GFLAG[F2]	eMIOS_0	eMIOS_0 channel 2 Flag
0x0360	54	eMIOS_0_GFLAG[F3]	eMIOS_0	eMIOS_0 channel 3 Flag
0x0370	55	eMIOS_0_GFLAG[F4]	eMIOS_0	eMIOS_0 channel 4 Flag

Table continues on the next page...

Table 4-4. Interrupt vectors and request sources (continued)

Offset	Vector	Source	Module	Description
0x0380	56	eMIOS_0_GFLAG[F5]	eMIOS_0	eMIOS_0 channel 5 Flag
0x0390	57	eMIOS_0_GFLAG[F6]	eMIOS_0	eMIOS_0 channel 6 Flag
0x03A0	58	eMIOS_0_GFLAG[F7]	eMIOS_0	eMIOS_0 channel 7 Flag
0x03B0	59	eMIOS_1_GFLAG[F0]	eMIOS_1	eMIOS_1 channel 0 Flag
0x03C0	60	eMIOS_1_GFLAG[F1]	eMIOS_1	eMIOS_1 channel 1 Flag
0x03D0	61	eMIOS_1_GFLAG[F2]	eMIOS_1	eMIOS_1 channel 2 Flag
0x03E0	62	eMIOS_1_GFLAG[F3]	eMIOS_1	eMIOS_1 channel 3 Flag
0x03F0	63	eMIOS_1_GFLAG[F4]	eMIOS_1	eMIOS_1 channel 4 Flag
0x0400	64	eMIOS_1_GFLAG[F5]	eMIOS_1	eMIOS_1 channel 5 Flag
0x0410	65	eMIOS_1_GFLAG[F6]	eMIOS_1	eMIOS_1 channel 6 Flag
0x0420	66	eMIOS_1_GFLAG[F7]	eMIOS_1	eMIOS_1 channel 7 Flag
0x0430	67	eTPU_SCR_MCR[SDMERR,SSAE RR,SCMERR,WDT01/2,MGE1/2,IL F1/2,LTO1/2,SCMMISF] and eTPU_MESR_MESR[DCERR,CCE RR]	eTPU_A/B	eTPU Engine A and B Global Exception
0x0440	68	eTPU_GCR_CISR_ENG1[CIS0]	eTPU_A	eTPU Engine A Channel 0 Interrupt Status
0x0450	69	eTPU_GCR_CISR_ENG1[CIS1]	eTPU_A	eTPU Engine A Channel 1 Interrupt Status
0x0460	70	eTPU_GCR_CISR_ENG1[CIS2]	eTPU_A	eTPU Engine A Channel 2 Interrupt Status
0x0470	71	eTPU_GCR_CISR_ENG1[CIS3]	eTPU_A	eTPU Engine A Channel 3 Interrupt Status
0x0480	72	eTPU_GCR_CISR_ENG1[CIS4]	eTPU_A	eTPU Engine A Channel 4 Interrupt Status
0x0490	73	eTPU_GCR_CISR_ENG1[CIS5]	eTPU_A	eTPU Engine A Channel 5 Interrupt Status
0x04A0	74	eTPU_GCR_CISR_ENG1[CIS6]	eTPU_A	eTPU Engine A Channel 6 Interrupt Status
0x04B0	75	eTPU_GCR_CISR_ENG1[CIS7]	eTPU_A	eTPU Engine A Channel 7 Interrupt Status
0x04C0	76	eTPU_GCR_CISR_ENG1[CIS8]	eTPU_A	eTPU Engine A Channel 8 Interrupt Status
0x04D0	77	eTPU_GCR_CISR_ENG1[CIS9]	eTPU_A	eTPU Engine A Channel 9 Interrupt Status
0x04E0	78	eTPU_GCR_CISR_ENG1[CIS10]	eTPU_A	eTPU Engine A Channel 10 Interrupt Status
0x04F0	79	eTPU_GCR_CISR_ENG1[CIS11]	eTPU_A	eTPU Engine A Channel 11 Interrupt Status
0x0500	80	eTPU_GCR_CISR_ENG1[CIS12]	eTPU_A	eTPU Engine A Channel 12 Interrupt Status
0x0510	81	eTPU_GCR_CISR_ENG1[CIS13]	eTPU_A	eTPU Engine A Channel 13 Interrupt Status
0x0520	82	eTPU_GCR_CISR_ENG1[CIS14]	eTPU_A	eTPU Engine A Channel 14 Interrupt Status
0x0530	83	eTPU_GCR_CISR_ENG1[CIS15]	eTPU_A	eTPU Engine A Channel 15 Interrupt Status
0x0540	84	eTPU_GCR_CISR_ENG1[CIS16]	eTPU_A	eTPU Engine A Channel 16 Interrupt Status
0x0550	85	eTPU_GCR_CISR_ENG1[CIS17]	eTPU_A	eTPU Engine A Channel 17 Interrupt Status
0x0560	86	eTPU_GCR_CISR_ENG1[CIS18]	eTPU_A	eTPU Engine A Channel 18 Interrupt Status
0x0570	87	eTPU_GCR_CISR_ENG1[CIS19]	eTPU_A	eTPU Engine A Channel 19 Interrupt Status
0x0580	88	eTPU_GCR_CISR_ENG1[CIS20]	eTPU_A	eTPU Engine A Channel 20 Interrupt Status
0x0590	89	eTPU_GCR_CISR_ENG1[CIS21]	eTPU_A	eTPU Engine A Channel 21 Interrupt Status
0x05A0	90	eTPU_GCR_CISR_ENG1[CIS22]	eTPU_A	eTPU Engine A Channel 22 Interrupt Status
0x05B0	91	eTPU_GCR_CISR_ENG1[CIS23]	eTPU_A	eTPU Engine A Channel 23 Interrupt Status
0x05C0	92	eTPU_GCR_CISR_ENG1[CIS24]	eTPU_A	eTPU Engine A Channel 24 Interrupt Status

Table continues on the next page...

Table 4-4. Interrupt vectors and request sources (continued)

Offset	Vector	Source	Module	Description
0x05D0	93	eTPU_GCR_CISR_ENG1[CIS25]	eTPU_A	eTPU Engine A Channel 25 Interrupt Status
0x05E0	94	eTPU_GCR_CISR_ENG1[CIS26]	eTPU_A	eTPU Engine A Channel 26 Interrupt Status
0x05F0	95	eTPU_GCR_CISR_ENG1[CIS27]	eTPU_A	eTPU Engine A Channel 27 Interrupt Status
0x0600	96	eTPU_GCR_CISR_ENG1[CIS28]	eTPU_A	eTPU Engine A Channel 28 Interrupt Status
0x0610	97	eTPU_GCR_CISR_ENG1[CIS29]	eTPU_A	eTPU Engine A Channel 29 Interrupt Status
0x0620	98	eTPU_GCR_CISR_ENG1[CIS30]	eTPU_A	eTPU Engine A Channel 30 Interrupt Status
0x0630	99	eTPU_GCR_CISR_ENG1[CIS31]	eTPU_A	eTPU Engine A Channel 31 Interrupt Status
0x0640	100	EQADCA_FISR _x [TORF] EQADCA_FISR _x [RFOF] EQADCA_FISR _x [CFUF]	eQADC_A	eQADC combined overrun interrupt requests from all of the FIFOs: Trigger Overrun, Receive FIFO Overflow, and command FIFO Underflow
0x0650	101	EQADCA_FISR0[NCF]	eQADC_A	eQADC command FIFO 0 Non-Coherency Flag
0x0660	102	EQADCA_FISR0[PF]	eQADC_A	eQADC command FIFO 0 Pause Flag
0x0670	103	EQADCA_FISR0[EOQF]	eQADC_A	eQADC command FIFO 0 command queue End of Queue Flag
0x0680	104	EQADCA_FISR0[CFFF]	eQADC_A	eQADC Command FIFO 0 Fill Flag
0x0690	105	EQADCA_FISR0[RFDF]	eQADC_A	eQADC Receive FIFO 0 Drain Flag
0x06A0	106	EQADCA_FISR1[NCF]	eQADC_A	eQADC command FIFO 1 Non-Coherency Flag
0x06B0	107	EQADCA_FISR1[PF]	eQADC_A	eQADC command FIFO 1 Pause Flag
0x06C0	108	EQADCA_FISR1[EOQF]	eQADC_A	eQADC command FIFO 1 command queue End of Queue Flag
0x06D0	109	EQADCA_FISR1[CFFF]	eQADC_A	eQADC Command FIFO 1 Fill Flag
0x06E0	110	EQADCA_FISR1[RFDF]	eQADC_A	eQADC Receive FIFO 1 Drain Flag
0x06F0	111	EQADCA_FISR2[NCF]	eQADC_A	eQADC command FIFO 2 Non-Coherency Flag
0x0700	112	EQADCA_FISR2[PF]	eQADC_A	eQADC command FIFO 2 Pause Flag
0x0710	113	EQADCA_FISR2[EOQF]	eQADC_A	eQADC command FIFO 2 command queue End of Queue Flag
0x0720	114	EQADCA_FISR2[CFFF]	eQADC_A	eQADC Command FIFO 2 Fill Flag
0x0730	115	EQADCA_FISR2[RFDF]	eQADC_A	eQADC Receive FIFO 2 Drain Flag
0x0740	116	EQADCA_FISR3[NCF]	eQADC_A	eQADC command FIFO 3 Non-Coherency Flag
0x0750	117	EQADCA_FISR3[PF]	eQADC_A	eQADC command FIFO 3 Pause Flag
0x0760	118	EQADCA_FISR3[EOQF]	eQADC_A	eQADC command FIFO 3 command queue End of Queue Flag
0x0770	119	EQADCA_FISR3[CFFF]	eQADC_A	eQADC Command FIFO 3 Fill Flag
0x0780	120	EQADCA_FISR3[RFDF]	eQADC_A	eQADC Receive FIFO 3 Drain Flag
0x0790	121	EQADCA_FISR4[NCF]	eQADC_A	eQADC command FIFO 4 Non-Coherency Flag
0x07A0	122	EQADCA_FISR4[PF]	eQADC_A	eQADC command FIFO 4 Pause Flag

Table continues on the next page...

Table 4-4. Interrupt vectors and request sources (continued)

Offset	Vector	Source	Module	Description
0x07B0	123	EQADCA_FISR4[EOQF]	eQADC_A	eQADC command FIFO 4 command queue End of Queue Flag
0x07C0	124	EQADCA_FISR4[CFFF]	eQADC_A	eQADC Command FIFO 4 Fill Flag
0x07D0	125	EQADCA_FISR4[RFDF]	eQADC_A	eQADC Receive FIFO 4 Drain Flag
0x07E0	126	EQADCA_FISR5[NCF]	eQADC_A	eQADC command FIFO 5 Non-Coherency Flag
0x07F0	127	EQADCA_FISR5[PF]	eQADC_A	eQADC command FIFO 5 Pause Flag
0x0800	128	EQADCA_FISR5[EOQF]	eQADC_A	eQADC command FIFO 5 command queue End of Queue Flag
0x0810	129	EQADCA_FISR5[CFFF]	eQADC_A	eQADC Command FIFO 5 Fill Flag
0x0820	130	EQADCA_FISR5[RFDF]	eQADC_A	eQADC Receive FIFO 5 Drain Flag
0x0830	131	DSPIB_SR[TFUF] DSPIB_SR[RFOF] DSPIB_SR[SPEF] DSPIB_SR[DPEF]	DSPI_B	DSPI_B combined overrun and parity error interrupt requests: Transmit FIFO Underflow/Receive FIFO Overflow SPI Parity Error/DSI Parity Error
0x0840	132	DSPIB_SR[EOQF]	DSPI_B	DSPI_B transmit FIFO End of Queue Flag
0x0850	133	DSPIB_SR[TFFF]	DSPI_B	DSPI_B Transmit FIFO Fill Flag
0x0860	134	DSPIB_SR[TCF] DSPIB_SR[DDIF]	DSPI_B	DSPI_B Transfer Complete/DSI Data Match Flag
0x0870	135	DSPIB_SR[RFDF]	DSPI_B	DSPI_B Receive FIFO Drain Flag
0x0880	136	DSPIC_SR[TFUF] DSPIC_SR[RFOF] DSPIC_SR[SPEF] DSPIC_SR[DPEF]	DSPI_C	DSPI_C combined overrun and parity error interrupt requests: Transmit FIFO Underflow/Receive FIFO Overflow SPI Parity Error/DSI Parity Error
0x0890	137	DSPIC_SR[EOQF]	DSPI_C	DSPI_C transmit FIFO End of Queue Flag
0x08A0	138	DSPIC_SR[TFFF]	DSPI_C	DSPI_C Transmit FIFO Fill Flag
0x08B0	139	DSPIC_SR[TCF] DSPIC_SR[DDIF]	DSPI_C	DSPI_C Transfer Complete/DSI Data Match Flag
0x08C0	140	DSPIC_SR[RFDF]	DSPI_C	DSPI_C Receive FIFO Drain Flag
0x08D0	141	DSPID_SR[TFUF] DSPID_SR[RFOF] DSPID_SR[SPEF] DSPID_SR[DPEF]	DSPI_D	DSPI_D combined overrun and parity error interrupt requests: Transmit FIFO Underflow/Receive FIFO Overflow SPI Parity Error/DSI Parity Error
0x08E0	142	DSPID_SR[EOQF]	DSPI_D	DSPI_D transmit FIFO End of Queue Flag
0x08F0	143	DSPID_SR[TFFF]	DSPI_D	DSPI_D Transmit FIFO Fill Flag
0x0900	144	DSPID_SR[TCF] DSPID_SR[DDIF]	DSPI_D	DSPI_D Transfer Complete/DSI Data Match Flag
0x0910	145	DSPID_SR[RFDF]	DSPI_D	DSPI_D Receive FIFO Drain Flag

Table continues on the next page...

Table 4-4. Interrupt vectors and request sources (continued)

Offset	Vector	Source	Module	Description
0x0920	146	ESCIA_SR[TDRE] ESCIA_SR[TC] ESCIA_SR[RDRF] ESCIA_SR[IDLE] ESCIA_SR[OR] ESCIA_SR[NF] ESCIA_SR[FE] ESCIA_SR[PF] ESCIA_SR[BERR] ESCIA_SR[RXRDY] ESCIA_SR[TXRDY] ESCIA_SR[LWAKE] ESCIA_SR[STO] ESCIA_SR[PBERR] ESCIA_SR[CERR] ESCIA_SR[CKERR] ESCIA_SR[FRC] ESCIA_SR[OVFL]	eSCI_A	Combined Interrupt Requests of ESCI Module A: Transmit Data Register Empty, Transmit Complete, Receive Data Register Full, Idle line, Overrun, Noise Flag, Framing Error Flag, and Parity Error Flag interrupt requests, SCI Status Register 2 Bit Error interrupt request, LIN Status Register 1 Receive Data Ready, Transmit Data Ready, Received LIN Wakeup Signal, Slave TimeOut, Physical Bus Error, CRC Error, Checksum Error, Frame Complete interrupts requests, and LIN Status Register 2 Receive Register Overflow
0x0930	147	Master0 Snoop ipi_int	PCU	PCU_IR0[OIF] PCU_IR0[EIF]
0x0940	148	Master1 Snoop ipi_int	PCU	PCU_IR1[OIF] PCU_IR1[EIF]

Table continues on the next page...

Table 4-4. Interrupt vectors and request sources (continued)

Offset	Vector	Source	Module	Description
0x0950	149	ESCIB_SR[TDRE] ESCIB_SR[TC] ESCIB_SR[RDRF] ESCIB_SR[IDLE] ESCIB_SR[OR] ESCIB_SR[NF] ESCIB_SR[FE] ESCIB_SR[PF] ESCIB_SR[BERR] ESCIB_SR[RXRDY] ESCIB_SR[TXRDY] ESCIB_SR[LWAKE] ESCIB_SR[STO] ESCIB_SR[PBERR] ESCIB_SR[CERR] ESCIB_SR[CKERR] ESCIB_SR[FRC] ESCIB_SR[OVFL]	eSCI_B	Combined Interrupt Requests of ESCI Module B: Transmit Data Register Empty, Transmit Complete, Receive Data Register Full, Idle line, Overrun, Noise Flag, Framing Error Flag, and Parity Error Flag interrupt requests, SCI Status Register 2 Bit Error interrupt request, LIN Status Register 1 Receive Data Ready, Transmit Data Ready, Received LIN Wakeup Signal, Slave TimeOut, Physical Bus Error, CRC Error, Checksum Error, Frame Complete interrupts requests, and LIN Status Register 2 Receive Register Overflow
0x0960	150	PSI5_0_CH_0_DSR[IS_DMA_TF_P M_DS] [IS_DMA_TF_SF] [IS_DMA_PM_DS_FIFO_FULL] [IS_DMA_SFUF] [IS_DMA_PM_DS_UF] NDSR[NDS31:0] OWSR[OWS31:0] EISR[ERROR31:0]	PSI5_0	PSI5_0 DMA Status, New data, OverWrite, Error interrupts
0x0970	151	PSI5_1_CH_0_DSR[IS_DMA_TF_P M_DS] [IS_DMA_TF_SF] [IS_DMA_PM_DS_FIFO_FULL] [IS_DMA_SFUF] [IS_DMA_PM_DS_UF] NDSR[NDS31:0] OWSR[OWS31:0] EISR[ERROR31:0]	PSI5_1	PSI5_1 DMA Status, New data, OverWrite, Error interrupts
0x0980	152	CANA_ESR1[BOFF_INT] [TWRNINT] [RWRNINT]	FlexCAN_A	FlexCAN_A Bus Off, Transmit Warning, Receive Warning
0x0990	153	CANA_ERRSR[ERR_INT] CANA_ERRSR[ECC_CE_INT] CANA_ERRSR[ECC_HANCE_INT] CANA_ERRSR[ECC_FANCE_INT]	FlexCAN_A	FlexCAN_A Error, FlexCAN_A ECC Correctable Error, FlexCAN_A ECC Host Access Non-Correctable Error, FlexCAN_A ECC CAN Access Non-Correctable Error
0x09A0	154	Reserved	FlexCAN_A	Reserved
0x09B0	155	CANA_IFRL[BUF0]	FlexCAN_A	FlexCAN_A Buffer 0 Interrupt
0x09C0	156	CANA_IFRL[BUF1]	FlexCAN_A	FlexCAN_A Buffer 1 Interrupt
0x09D0	157	CANA_IFRL[BUF2]	FlexCAN_A	FlexCAN_A Buffer 2 Interrupt
0x09E0	158	CANA_IFRL[BUF3]	FlexCAN_A	FlexCAN_A Buffer 3 Interrupt

Table continues on the next page...

Table 4-4. Interrupt vectors and request sources (continued)

Offset	Vector	Source	Module	Description
0x09F0	159	CANA_IFRL[BUF4]	FlexCAN_A	FlexCAN_A Buffer 4 Interrupt
0x0A00	160	CANA_IFRL[BUF5]	FlexCAN_A	FlexCAN_A Buffer 5 Interrupt
0x0A10	161	CANA_IFRL[BUF6]	FlexCAN_A	FlexCAN_A Buffer 6 Interrupt
0x0A20	162	CANA_IFRL[BUF7]	FlexCAN_A	FlexCAN_A Buffer 7 Interrupt
0x0A30	163	CANA_IFRL[BUF8]	FlexCAN_A	FlexCAN_A Buffer 8 Interrupt
0x0A40	164	CANA_IFRL[BUF9]	FlexCAN_A	FlexCAN_A Buffer 9 Interrupt
0x0A50	165	CANA_IFRL[BUF10]	FlexCAN_A	FlexCAN_A Buffer 10 Interrupt
0x0A60	166	CANA_IFRL[BUF11]	FlexCAN_A	FlexCAN_A Buffer 11 Interrupt
0x0A70	167	CANA_IFRL[BUF12]	FlexCAN_A	FlexCAN_A Buffer 12 Interrupt
0x0A80	168	CANA_IFRL[BUF13]	FlexCAN_A	FlexCAN_A Buffer 13 Interrupt
0x0A90	169	CANA_IFRL[BUF14]	FlexCAN_A	FlexCAN_A Buffer 14 Interrupt
0x0AA0	170	CANA_IFRL[BUF15]	FlexCAN_A	FlexCAN_A Buffer 15 Interrupt
0x0AB0	171	CANA_IFRL[BUF31:BUF16]	FlexCAN_A	FlexCAN_A Buffers 31-16 Interrupts
0x0AC0	172	CANA_IFRH[BUF63:BUF32]	FlexCAN_A	FlexCAN_A Buffers 63-32 Interrupts
0x0AD0	173	CANC_ESR1[BOFF_INT] [TWRNINT] [RWRNINT]	FlexCAN_C	FlexCAN_C Bus Off, Transmit Warning, Receive Warning
0x0AE0	174	CANC_ESR1[ERR_INT] CANC_ERRSR[ECC_CE_INT] CANC_ERRSR[ECC_HANCE_INT] CANC_ERRSR[ECC_FANCE_INT]	FlexCAN_C	FlexCAN_C Error, FlexCAN_C ECC Correctable Error, FlexCAN_C ECC Host Access Non-Correctable Error, FlexCAN_C ECC CAN Access Non-Correctable Error
0x0AF0	175	Reserved	FlexCAN_C	Reserved
0x0B00	176	CANC_IFRL[BUF0]	FlexCAN_C	FlexCAN_C Buffer 0 Interrupt
0x0B10	177	CANC_IFRL[BUF1]	FlexCAN_C	FlexCAN_C Buffer 1 Interrupt
0x0B20	178	CANC_IFRL[BUF2]	FlexCAN_C	FlexCAN_C Buffer 2 Interrupt
0x0B30	179	CANC_IFRL[BUF3]	FlexCAN_C	FlexCAN_C Buffer 3 Interrupt
0x0B40	180	CANC_IFRL[BUF4]	FlexCAN_C	FlexCAN_C Buffer 4 Interrupt
0x0B50	181	CANC_IFRL[BUF5]	FlexCAN_C	FlexCAN_C Buffer 5 Interrupt
0x0B60	182	CANC_IFRL[BUF6]	FlexCAN_C	FlexCAN_C Buffer 6 Interrupt
0x0B70	183	CANC_IFRL[BUF7]	FlexCAN_C	FlexCAN_C Buffer 7 Interrupt
0x0B80	184	CANC_IFRL[BUF8]	FlexCAN_C	FlexCAN_C Buffer 8 Interrupt
0x0B90	185	CANC_IFRL[BUF9]	FlexCAN_C	FlexCAN_C Buffer 9 Interrupt
0x0BA0	186	CANC_IFRL[BUF10]	FlexCAN_C	FlexCAN_C Buffer 10 Interrupt
0x0BB0	187	CANC_IFRL[BUF11]	FlexCAN_C	FlexCAN_C Buffer 11 Interrupt
0x0BC0	188	CANC_IFRL[BUF12]	FlexCAN_C	FlexCAN_C Buffer 12 Interrupt
0x0BD0	189	CANC_IFRL[BUF13]	FlexCAN_C	FlexCAN_C Buffer 13 Interrupt
0x0BE0	190	CANC_IFRL[BUF14]	FlexCAN_C	FlexCAN_C Buffer 14 Interrupt
0x0BF0	191	CANC_IFRL[BUF15]	FlexCAN_C	FlexCAN_C Buffer 15 Interrupt
0x0C00	192	CANC_IFRL[BUF31:BUF16]	FlexCAN_C	FlexCAN_C Buffers 31-16 Interrupts
0x0C10	193	CANC_IFRH[BUF63:BUF32]	FlexCAN_C	FlexCAN_C Buffers 63-32 Interrupts
0x0C20	194	EIR[TXF]	FEC	FEC Transmit Frame flag
0x0C30	195	EIR[RXF]	FEC	FEC Receive Frame flag

Table continues on the next page...

Table 4-4. Interrupt vectors and request sources (continued)

Offset	Vector	Source	Module	Description
0x0C40	196	EIR[HBERR] EIR[BABR] EIR[BABT] EIR[GRA] EIR[TXB] EIR[RXB] EIR[MII] EIR[EBERR] EIR[LC] EIR[RL] EIR[UN]	FEC	Combined Interrupt Requests of the FEC Ethernet Interrupt Event Register: Heartbeat Error, Babbling Receive Error, Babbling Transmit Error, Graceful Stop Complete, Transmit Buffer, Receive Buffer, Media Independent Interface, Ethernet Bus Error, Late Collision, Collision Retry Limit, and Transmit FIFO Underrun
0x0C50	197	DECFILTER_MSR_A[IDF]	DEC_A	Decimation A Input (Fill)
0x0C60	198	DECFILTER_MSR_A[ODFI SDF]	DEC_A	Decimation A Output/Integ (Drain/Integ)
0x0C70	199	DECFILTER A ERRORS	DEC_A	Decimation A Error
0x0C80	200	STM_CIR0[CIF]	STM	System Timer Module Interrupt 0
0x0C90	201	STM_CIR1[CIF], STM_CIR2[CIF], STM_CIR3[CIF]	STM	System Timer Module Interrupts 1, 2, 3
0x0CA0	202	eMIOS_0_GFLAG[F16]	eMIOS_0	eMIOS_0 channel 16 Flag
0x0CB0	203	eMIOS_0_GFLAG[F17]	eMIOS_0	eMIOS_0 channel 17 Flag
0x0CC0	204	eMIOS_0_GFLAG[F18]	eMIOS_0	eMIOS_0 channel 18 Flag
0x0CD0	205	eMIOS_0_GFLAG[F19]	eMIOS_0	eMIOS_0 channel 19 Flag
0x0CE0	206	eMIOS_0_GFLAG[F20]	eMIOS_0	eMIOS_0 channel 20 Flag
0x0CF0	207	eMIOS_0_GFLAG[F21]	eMIOS_0	eMIOS_0 channel 21 Flag
0x0D00	208	eMIOS_0_GFLAG[F22]	eMIOS_0	eMIOS_0 channel 22 Flag
0x0D10	209	eMIOS_0_GFLAG[F23]	eMIOS_0	eMIOS_0 channel 23 Flag
0x0D20	210	EDMAA_ERRH[ERR63:ERR32]	eDMA_A	eDMA channel Error flags 32-63
0x0D30	211	EDMAA_IRQRH[INT32]	eDMA_A	eDMA channel Interrupt 32
0x0D40	212	EDMAA_IRQRH[INT33]	eDMA_A	eDMA channel Interrupt 33
0x0D50	213	EDMAA_IRQRH[INT34]	eDMA_A	eDMA channel Interrupt 34
0x0D60	214	EDMAA_IRQRH[INT35]	eDMA_A	eDMA channel Interrupt 35
0x0D70	215	EDMAA_IRQRH[INT36]	eDMA_A	eDMA channel Interrupt 36
0x0D80	216	EDMAA_IRQRH[INT37]	eDMA_A	eDMA channel Interrupt 37
0x0D90	217	EDMAA_IRQRH[INT38]	eDMA_A	eDMA channel Interrupt 38
0x0DA0	218	EDMAA_IRQRH[INT39]	eDMA_A	eDMA channel Interrupt 39
0x0DB0	219	EDMAA_IRQRH[INT40]	eDMA_A	eDMA channel Interrupt 40
0x0DC0	220	EDMAA_IRQRH[INT41]	eDMA_A	eDMA channel Interrupt 41
0x0DD0	221	EDMAA_IRQRH[INT42]	eDMA_A	eDMA channel Interrupt 42
0x0DE0	222	EDMAA_IRQRH[INT43]	eDMA_A	eDMA channel Interrupt 43
0x0DF0	223	EDMAA_IRQRH[INT44]	eDMA_A	eDMA channel Interrupt 44
0x0E00	224	EDMAA_IRQRH[INT45]	eDMA_A	eDMA channel Interrupt 45
0x0E10	225	EDMAA_IRQRH[INT46]	eDMA_A	eDMA channel Interrupt 46
0x0E20	226	EDMAA_IRQRH[INT47]	eDMA_A	eDMA channel Interrupt 47
0x0E30	227	EDMAA_IRQRH[INT48]	eDMA_A	eDMA channel Interrupt 48
0x0E40	228	EDMAA_IRQRH[INT49]	eDMA_A	eDMA channel Interrupt 49

Table continues on the next page...

Table 4-4. Interrupt vectors and request sources (continued)

Offset	Vector	Source	Module	Description
0x0E50	229	EDMAA_IRQRH[INT50]	eDMA_A	eDMA channel Interrupt 50
0x0E60	230	EDMAA_IRQRH[INT51]	eDMA_A	eDMA channel Interrupt 51
0x0E70	231	EDMAA_IRQRH[INT52]	eDMA_A	eDMA channel Interrupt 52
0x0E80	232	EDMAA_IRQRH[INT53]	eDMA_A	eDMA channel Interrupt 53
0x0E90	233	EDMAA_IRQRH[INT54]	eDMA_A	eDMA channel Interrupt 54
0x0EA0	234	EDMAA_IRQRH[INT55]	eDMA_A	eDMA channel Interrupt 55
0x0EB0	235	EDMAA_IRQRH[INT56]	eDMA_A	eDMA channel Interrupt 56
0x0EC0	236	EDMAA_IRQRH[INT57]	eDMA_A	eDMA channel Interrupt 57
0x0ED0	237	EDMAA_IRQRH[INT58]	eDMA_A	eDMA channel Interrupt 58
0x0EE0	238	EDMAA_IRQRH[INT59]	eDMA_A	eDMA channel Interrupt 59
0x0EF0	239	EDMAA_IRQRH[INT60]	eDMA_A	eDMA channel Interrupt 60
0x0F00	240	EDMAA_IRQRH[INT61]	eDMA_A	eDMA channel Interrupt 61
0x0F10	241	EDMAA_IRQRH[INT62]	eDMA_A	eDMA channel Interrupt 62
0x0F20	242	EDMAA_IRQRH[INT63]	eDMA_A	eDMA channel Interrupt 63
0x0F30	243	eTPU_GCR_CISR_ENG2[CIS0]	eTPU_B	eTPU Engine B Channel 0 Interrupt Status
0x0F40	244	eTPU_GCR_CISR_ENG2[CIS1]	eTPU_B	eTPU Engine B Channel 1 Interrupt Status
0x0F50	245	eTPU_GCR_CISR_ENG2[CIS2]	eTPU_B	eTPU Engine B Channel 2 Interrupt Status
0x0F60	246	eTPU_GCR_CISR_ENG2[CIS3]	eTPU_B	eTPU Engine B Channel 3 Interrupt Status
0x0F70	247	eTPU_GCR_CISR_ENG2[CIS4]	eTPU_B	eTPU Engine B Channel 4 Interrupt Status
0x0F80	248	eTPU_GCR_CISR_ENG2[CIS5]	eTPU_B	eTPU Engine B Channel 5 Interrupt Status
0x0F90	249	eTPU_GCR_CISR_ENG2[CIS6]	eTPU_B	eTPU Engine B Channel 6 Interrupt Status
0x0FA0	250	eTPU_GCR_CISR_ENG2[CIS7]	eTPU_B	eTPU Engine B Channel 7 Interrupt Status
0x0FB0	251	eTPU_GCR_CISR_ENG2[CIS8]	eTPU_B	eTPU Engine B Channel 8 Interrupt Status
0x0FC0	252	eTPU_GCR_CISR_ENG2[CIS9]	eTPU_B	eTPU Engine B Channel 9 Interrupt Status
0x0fd0	253	eTPU_GCR_CISR_ENG2[CIS10]	eTPU_B	eTPU Engine B Channel 10 Interrupt Status
0x0fe0	254	eTPU_GCR_CISR_ENG2[CIS11]	eTPU_B	eTPU Engine B Channel 11 Interrupt Status
0x0ff0	255	eTPU_GCR_CISR_ENG2[CIS12]	eTPU_B	eTPU Engine B Channel 12 Interrupt Status
0x1000	256	eTPU_GCR_CISR_ENG2[CIS13]	eTPU_B	eTPU Engine B Channel 13 Interrupt Status
0x1010	257	eTPU_GCR_CISR_ENG2[CIS14]	eTPU_B	eTPU Engine B Channel 14 Interrupt Status
0x1020	258	eTPU_GCR_CISR_ENG2[CIS15]	eTPU_B	eTPU Engine B Channel 15 Interrupt Status
0x1030	259	eTPU_GCR_CISR_ENG2[CIS16]	eTPU_B	eTPU Engine B Channel 16 Interrupt Status
0x1040	260	eTPU_GCR_CISR_ENG2[CIS17]	eTPU_B	eTPU Engine B Channel 17 Interrupt Status
0x1050	261	eTPU_GCR_CISR_ENG2[CIS18]	eTPU_B	eTPU Engine B Channel 18 Interrupt Status
0x1060	262	eTPU_GCR_CISR_ENG2[CIS19]	eTPU_B	eTPU Engine B Channel 19 Interrupt Status
0x1070	263	eTPU_GCR_CISR_ENG2[CIS20]	eTPU_B	eTPU Engine B Channel 20 Interrupt Status
0x1080	264	eTPU_GCR_CISR_ENG2[CIS21]	eTPU_B	eTPU Engine B Channel 21 Interrupt Status
0x1090	265	eTPU_GCR_CISR_ENG2[CIS22]	eTPU_B	eTPU Engine B Channel 22 Interrupt Status
0x10A0	266	eTPU_GCR_CISR_ENG2[CIS23]	eTPU_B	eTPU Engine B Channel 23 Interrupt Status
0x10B0	267	eTPU_GCR_CISR_ENG2[CIS24]	eTPU_B	eTPU Engine B Channel 24 Interrupt Status

Table continues on the next page...

Table 4-4. Interrupt vectors and request sources (continued)

Offset	Vector	Source	Module	Description
0x10C0	268	eTPU_GCR_CISR_ENG2[CIS25]	eTPU_B	eTPU Engine B Channel 25 Interrupt Status
0x10D0	269	eTPU_GCR_CISR_ENG2[CIS26]	eTPU_B	eTPU Engine B Channel 26 Interrupt Status
0x10E0	270	eTPU_GCR_CISR_ENG2[CIS27]	eTPU_B	eTPU Engine B Channel 27 Interrupt Status
0x10F0	271	eTPU_GCR_CISR_ENG2[CIS28]	eTPU_B	eTPU Engine B Channel 28 Interrupt Status
0x1100	272	eTPU_GCR_CISR_ENG2[CIS29]	eTPU_B	eTPU Engine B Channel 29 Interrupt Status
0x1110	273	eTPU_GCR_CISR_ENG2[CIS30]	eTPU_B	eTPU Engine B Channel 30 Interrupt Status
0x1120	274	eTPU_GCR_CISR_ENG2[CIS31]	eTPU_B	eTPU Engine B Channel 31 Interrupt Status
0x1130	275	DSPIA_SR[TFUF] DSPIA_SR[RFOF] DSPIA_SR[SPEF] DSPIA_SR[DPEF]	DSPI_A	DSPI_A combined overrun and parity error interrupt requests: Transmit FIFO Underflow/Receive FIFO Overflow SPI Parity Error/DSI Parity Error
0x1140	276	DSPIA_SR[EOQF]	DSPI_A	DSPI_A transmit FIFO End of Queue Flag
0x1150	277	DSPIA_SR[TFFF]	DSPI_A	DSPI_A Transmit FIFO Fill Flag
0x1160	278	DSPIA_SR[TCF] DSPIA_SR[DDIF]	DSPI_A	DSPI_A Transfer Complete/DSI Data Match Flag
0x1170	279	DSPIA_SR[RFDF]	DSPI_A	DSPI_A Receive FIFO Drain Flag
0x1180	280	CANB_ESR1[BOFF_INT] [TWRNINT] [RWRNINT]	FlexCAN_B	FlexCAN_B Bus Off, Transmit Warning, Receive Warning
0x1190	281	CANB_ESR1[ERR_INT] CANB_ERRSR[ECC_CE_INT] CANB_ERRSR[ECC_HANCE_INT] CANB_ERRSR[ECC_FANCE_INT]	FlexCAN_B	FlexCAN_B Error, FlexCAN_B ECC Correctable Error, FlexCAN_B ECC Host Access Non-Correctable Error, FlexCAN_B ECC CAN Access Non-Correctable Error
0x11A0	282	Reserved	FlexCAN_B	Reserved
0x11B0	283	CANB_IFRL[BUF0]	FlexCAN_B	FlexCAN_B Buffer 0 Interrupt
0x11C0	284	CANB_IFRL[BUF1]	FlexCAN_B	FlexCAN_B Buffer 1 Interrupt
0x11D0	285	CANB_IFRL[BUF2]	FlexCAN_B	FlexCAN_B Buffer 2 Interrupt
0x11E0	286	CANB_IFRL[BUF3]	FlexCAN_B	FlexCAN_B Buffer 3 Interrupt
0x11F0	287	CANB_IFRL[BUF4]	FlexCAN_B	FlexCAN_B Buffer 4 Interrupt
0x1200	288	CANB_IFRL[BUF5]	FlexCAN_B	FlexCAN_B Buffer 5 Interrupt
0x1210	289	CANB_IFRL[BUF6]	FlexCAN_B	FlexCAN_B Buffer 6 Interrupt
0x1220	290	CANB_IFRL[BUF7]	FlexCAN_B	FlexCAN_B Buffer 7 Interrupt
0x1230	291	CANB_IFRL[BUF8]	FlexCAN_B	FlexCAN_B Buffer 8 Interrupt
0x1240	292	CANB_IFRL[BUF9]	FlexCAN_B	FlexCAN_B Buffer 9 Interrupt
0x1250	293	CANB_IFRL[BUF10]	FlexCAN_B	FlexCAN_B Buffer 10 Interrupt
0x1260	294	CANB_IFRL[BUF11]	FlexCAN_B	FlexCAN_B Buffer 11 Interrupt
0x1270	295	CANB_IFRL[BUF12]	FlexCAN_B	FlexCAN_B Buffer 12 Interrupt
0x1280	296	CANB_IFRL[BUF13]	FlexCAN_B	FlexCAN_B Buffer 13 Interrupt
0x1290	297	CANB_IFRL[BUF14]	FlexCAN_B	FlexCAN_B Buffer 14 Interrupt
0x12A0	298	CANB_IFRL[BUF15]	FlexCAN_B	FlexCAN_B Buffer 15 Interrupt
0x12B0	299	CANB_IFRL[BUF31:BUF16]	FlexCAN_B	FlexCAN_B Buffers 31-16 Interrupts
0x12C0	300	CANB_IFRH[BUF63:BUF32]	FlexCAN_B	FlexCAN_B Buffers 63-32 Interrupts

Table continues on the next page...

Table 4-4. Interrupt vectors and request sources (continued)

Offset	Vector	Source	Module	Description
0x12D0	301	PIT_TFLG0[TIF]	PIT_RTI	Periodic Interrupt Timer Interrupt 0
0x12E0	302	PIT_TFLG1[TIF]	PIT_RTI	Periodic Interrupt Timer Interrupt 1
0x12F0	303	PIT_TFLG2[TIF]	PIT_RTI	Periodic Interrupt Timer Interrupt 2
0x1300	304	PIT_TFLG3[TIF]	PIT_RTI	Periodic Interrupt Timer Interrupt 3
0x1310	305	PIT_RTI_TFLG[TIF]	PIT_RTI	Real Time Interrupt Interrupt
0x1320	306	Reserved	—	Reserved
0x1330	307	MCR[DONE]	C55FMC	Flash memory program/erase complete
0x1340	308	CAND_ESR1[BOFF_INT] [TWRNINT] [RWRNINT]	FlexCAN_D	FlexCAN_D Bus Off, Transmit Warning, Receive Warning
0x1350	309	CAND_ESR1[ERR_INT] CAND_ERRSR[ECC_CE_INT] CAND_ERRSR[ECC_HANCE_INT] CAND_ERRSR[ECC_FANCE_INT]	FlexCAN_D	FlexCAN_D Error, FlexCAN_D ECC Correctable Error, FlexCAN_D ECC Host Access Non-Correctable Error, FlexCAN_D ECC CAN Access Non-Correctable Error
0x1360	310	Reserved	FlexCAN_D	Reserved
0x1370	311	CAND_IFRL[BUF0]	FlexCAN_D	FlexCAN_D buffer 0
0x1380	312	CAND_IFRL[BUF1]	FlexCAN_D	FlexCAN_D buffer 1
0x1390	313	CAND_IFRL[BUF2]	FlexCAN_D	FlexCAN_D buffer 2
0x13A0	314	CAND_IFRL[BUF3]	FlexCAN_D	FlexCAN_D buffer 3
0x13B0	315	CAND_IFRL[BUF4]	FlexCAN_D	FlexCAN_D buffer 4
0x13C0	316	CAND_IFRL[BUF5]	FlexCAN_D	FlexCAN_D buffer 5
0x13D0	317	CAND_IFRL[BUF6]	FlexCAN_D	FlexCAN_D buffer 6
0x13E0	318	CAND_IFRL[BUF7]	FlexCAN_D	FlexCAN_D buffer 7
0x13F0	319	CAND_IFRL[BUF8]	FlexCAN_D	FlexCAN_D buffer 8
0x1400	320	CAND_IFRL[BUF9]	FlexCAN_D	FlexCAN_D buffer 9
0x1410	321	CAND_IFRL[BUF10]	FlexCAN_D	FlexCAN_D buffer 10
0x1420	322	CAND_IFRL[BUF11]	FlexCAN_D	FlexCAN_D buffer 11
0x1430	323	CAND_IFRL[BUF12]	FlexCAN_D	FlexCAN_D buffer 12
0x1440	324	CAND_IFRL[BUF13]	FlexCAN_D	FlexCAN_D buffer 13
0x1450	325	CAND_IFRL[BUF14]	FlexCAN_D	FlexCAN_D buffer 14
0x1460	326	CAND_IFRL[BUF15]	FlexCAN_D	FlexCAN_D buffer 15
0x1470	327	CAND_IFRL[BUF31:BUF16]	FlexCAN_D	FlexCAN_D buffer 16-31
0x1480	328	CAND_IFRH[BUF63:BUF32]	FlexCAN_D	FlexCAN_D buffer 32-63
0x1490	329	SENT_0_GBL_STATUS[FMDU] SENT_0_GBL_STATUS[SMDU] SENT_0_GBL_STATUS[FMFO] all sources of vectors 330-335	SRX_0	SENT_0 Module Interrupts

Table continues on the next page...

Table 4-4. Interrupt vectors and request sources (continued)

Offset	Vector	Source	Module	Description
0x14A0	330	SENT_0_CH0_STATUS[CAL_RES YNCI CAL_20_50ISMSG_OFLWI FMSG_OFLWI NUM_EDGES_ERRI FMSG_CRC_ERRI SMSG_CRC_ERRINIB_VAL_ERRI CAL_DIAG_ERRICAL_LEN_ERRI PP_DIAG_ERR] I SENT_0_FMSG_RDY[0] I SENT_0_SMSG_RDY[0]	SRX_0	SENT_0_CH0 Interrupts
0x14B0	331	SENT_0_CH1_STATUS[CAL_RES YNCI CAL_20_50ISMSG_OFLWI FMSG_OFLWI NUM_EDGES_ERRI FMSG_CRC_ERRI SMSG_CRC_ERRINIB_VAL_ERRI CAL_DIAG_ERRICAL_LEN_ERRI PP_DIAG_ERR] I SENT_0_FMSG_RDY[1] I SENT_0_SMSG_RDY[1]	SRX_0	SENT_0_CH1 Interrupts
0x14C0	332	SENT_0_CH2_STATUS[CAL_RES YNCI CAL_20_50ISMSG_OFLWI FMSG_OFLWI NUM_EDGES_ERRI FMSG_CRC_ERRI SMSG_CRC_ERRINIB_VAL_ERRI CAL_DIAG_ERRICAL_LEN_ERRI PP_DIAG_ERR] I SENT_0_FMSG_RDY[2] I SENT_0_SMSG_RDY[2]	SRX_0	SENT_0_CH2 Interrupts
0x14D0	333	SENT_0_CH3_STATUS[CAL_RES YNCI CAL_20_50ISMSG_OFLWI FMSG_OFLWI NUM_EDGES_ERRI FMSG_CRC_ERRI SMSG_CRC_ERRINIB_VAL_ERRI CAL_DIAG_ERRICAL_LEN_ERRI PP_DIAG_ERR] I SENT_0_FMSG_RDY[3] I SENT_0_SMSG_RDY[3]	SRX_0	SENT_0_CH3 Interrupts
0x14E0	334	SENT_0_CH4_STATUS[CAL_RES YNCI CAL_20_50ISMSG_OFLWI FMSG_OFLWI NUM_EDGES_ERRI FMSG_CRC_ERRI SMSG_CRC_ERRINIB_VAL_ERRI CAL_DIAG_ERRICAL_LEN_ERRI PP_DIAG_ERR] I SENT_0_FMSG_RDY[4] I SENT_0_SMSG_RDY[4]	SRX_0	SENT_0_CH4 Interrupts

Table continues on the next page...

Table 4-4. Interrupt vectors and request sources (continued)

Offset	Vector	Source	Module	Description
0x14F0	335	SENT_0_CH5_STATUS[CAL_RES YNCI CAL_20_50ISMSG_OFLWI FMSG_OFLWI NUM_EDGES_ERRI FMSG_CRC_ERRI SMSG_CRC_ERRINIB_VAL_ERRI CAL_DIAG_ERRICAL_LEN_ERRI PP_DIAG_ERR] SENT_0_FMSG_RDY[5] SENT_0_SMSG_RDY[5]	SRX_0	SENT_0_CH5 Interrupts
0x1500	336	SENT_1_GBL_STATUS[FMDU] SENT_1_GBL_STATUS[SMDU] SENT_1_GBL_STATUS[FMFO] all sources of vectors 337-342	SRX_1	SENT_1 Module Interrupts
0x1510	337	SENT_1_CH0_STATUS[CAL_RES YNCI CAL_20_50ISMSG_OFLWI FMSG_OFLWI NUM_EDGES_ERRI FMSG_CRC_ERRI SMSG_CRC_ERRINIB_VAL_ERRI CAL_DIAG_ERRICAL_LEN_ERRI PP_DIAG_ERR] SENT_1_FMSG_RDY[0] SENT_1_SMSG_RDY[0]	SRX_1	SENT_1_CH0 Interrupts
0x1520	338	SENT_1_CH1_STATUS[CAL_RES YNCI CAL_20_50ISMSG_OFLWI FMSG_OFLWI NUM_EDGES_ERRI FMSG_CRC_ERRI SMSG_CRC_ERRINIB_VAL_ERRI CAL_DIAG_ERRICAL_LEN_ERRI PP_DIAG_ERR] SENT_1_FMSG_RDY[1] SENT_1_SMSG_RDY[1]	SRX_1	SENT_1_CH1 Interrupts
0x1530	339	SENT_1_CH2_STATUS[CAL_RES YNCI CAL_20_50ISMSG_OFLWI FMSG_OFLWI NUM_EDGES_ERRI FMSG_CRC_ERRI SMSG_CRC_ERRINIB_VAL_ERRI CAL_DIAG_ERRICAL_LEN_ERRI PP_DIAG_ERR] SENT_1_FMSG_RDY[2] SENT_1_SMSG_RDY[2]	SRX_1	SENT_1_CH2 Interrupts
0x1540	340	SENT_1_CH3_STATUS[CAL_RES YNCI CAL_20_50ISMSG_OFLWI FMSG_OFLWI NUM_EDGES_ERRI FMSG_CRC_ERRI SMSG_CRC_ERRINIB_VAL_ERRI CAL_DIAG_ERRICAL_LEN_ERRI PP_DIAG_ERR] SENT_1_FMSG_RDY[3] SENT_1_SMSG_RDY[3]	SRX_1	SENT_1_CH3 Interrupts

Table continues on the next page...

Table 4-4. Interrupt vectors and request sources (continued)

Offset	Vector	Source	Module	Description
0x1550	341	SENT_1_CH4_STATUS[CAL_RES YNCI CAL_20_50ISMSG_OFLWI FMSG_OFLWI NUM_EDGES_ERRI FMSG_CRC_ERRI SMSG_CRC_ERRINIB_VAL_ERRI CAL_DIAG_ERRICAL_LEN_ERRI PP_DIAG_ERR] SENT_1_FMSG_RDY[4] SENT_1_SMSG_RDY[4]	SRX_1	SENT_1_CH4 Interrupts
0x1560	342	SENT_1_CH5_STATUS[CAL_RES YNCI CAL_20_50ISMSG_OFLWI FMSG_OFLWI NUM_EDGES_ERRI FMSG_CRC_ERRI SMSG_CRC_ERRINIB_VAL_ERRI CAL_DIAG_ERRICAL_LEN_ERRI PP_DIAG_ERR] SENT_1_FMSG_RDY[5] SENT_1_SMSG_RDY[5]	SRX_1	SENT_1_CH5 Interrupts
0x1570	343	HVD_FLASH, HVD_HV, HVD_CORE, LVD_CORE_COLD of PMC_LVD_HVD_EVENT_STATUS	PMC	Power Management Controller Interrupts
0x1580	344	TEMP0_0, TEMP0_2, TEMP0_3, TEMP1_0, TEMP1_2, TEMP1_3 of PMC_ESR_TD	PMC Temp	Temperature Sensor Interrupts
0x1590	345	JDC_MSR[JIN_INT] JDC_MSR[JOUT_INT]	JDC	JDC Interrupts
0x15A0	346	SIPI_ERR SIPI_SR SIPI_CSR0	SIPI_0	SIPI Combined Interrupts
0x15B0	347	LFAST_0_TISR LFAST_0_RISR LFAST_0_RIISR	LFAST	LFAST Combined Interrupts
0x15C0	348	m_can0_int0 m_can0_int1 m_can1_int0 m_can1_int1	M_CAN	M_CAN0_0, M_CAN0_1, M_CAN1_0, M_CAN1_1 Combined Interrupts
0x15D0	349	Single bit Correction Multi bit Detection	ERM	ERM Combined Interrupts
0x15E0	350	CMU_0_ISR[FHHIIFLLIOLRI] CMU_1_ISR[FHHIIFLLI]	CMU	CMU_0, CMU_1 Clock Error Interrupts
0x15F0	351	CMU_2_ISR[FHHIIFLLI] CMU_3_ISR[FHHIIFLLI]	CMU	CMU_2, CMU_3 Clock Error Interrupts
0x1600	352	CMU_4_ISR[FHHIIFLLI] CMU_5_ISR[FHHIIFLLI]	CMU	CMU_4, CMU_5 Clock Error Interrupts
0x1610	353	CMU_6_ISR[FHHIIFLLI] CMU_7_ISR[FHHIIFLLI]	CMU	CMU_6, CMU_7 Clock Error Interrupts
0x1620	354	CMU_8_ISR[FHHIIFLLI]	CMU	CMU_8 Clock Error Interrupts
0x1630	355	Reserved	CMU	Reserved
0x1640	356	Reserved	CMU	Reserved
0x1650	357	Reserved	CMU	Reserved
0x1660	358	REACM_GEFR[OVRIEF7:0]	Reaction Channels	Reaction Module Global Interrupt

Table continues on the next page...

Table 4-4. Interrupt vectors and request sources (continued)

Offset	Vector	Source	Module	Description
0x1670	359	REACM_CHSR_0[MAXLIOCDFI SCDFITAERISQER] I REACM_CHSR_1[MAXLIOCDFI SCDFITAERISQER]	Reaction Channels	Reaction Channel 0 and Reaction Channel 1 Combined Interrupts
0x1680	360	REACM_CHSR_2[MAXLIOCDFI SCDFITAERISQER] I REACM_CHSR_3[MAXLIOCDFI SCDFITAERISQER]	Reaction Channels	Reaction Channel 2 and Reaction Channel 3 Combined Interrupts
0x1690	361	REACM_CHSR_4[MAXLIOCDFI SCDFITAERISQER] I REACM_CHSR_5[MAXLIOCDFI SCDFITAERISQER]	Reaction Channels	Reaction Channel 4 and Reaction Channel 5 Combined Interrupts
0x16A0	362	REACM_CHSR_6[MAXLIOCDFI SCDFITAERISQER] I REACM_CHSR_7[MAXLIOCDFI SCDFITAERISQER]	Reaction Channels	Reaction Channel 6 and Reaction Channel 7 Combined Interrupts
0x16B0	363	REACM_CHSR_8[MAXLIOCDFI SCDFITAERISQER] I REACM_CHSR_9[MAXLIOCDFI SCDFITAERISQER]	Reaction Channels	Reaction Channel 8 and Reaction Channel 9 Combined Interrupts
0x16C0	364	Reserved	Reaction Channels	Reserved
0x16D0	365	Reserved	Reaction Channels	Reserved
0x16E0	366	DECFILTER_MSR_B[IDF]	DEC_B	Decimation B Input (Fill)
0x16F0	367	DECFILTER_MSR_B[ODFI SDF]	DEC_B	Decimation B Output/Integ (Drain/Integ)
0x1700	368	DECFILTER B ERRORS	DEC_B	Decimation B Error
0x1710	369	eTPU_SCR_MCR[SDMERR,SSAE RR,SCMERR,WDT01,MGE1,ILF1,L TO1,SCMMISF] and eTPU_MESR_MESR[DCERR,CCE RR]	eTPU_C	eTPU Engine C Global Exception
0x1720	370	eTPU_GCR_CISR_ENG1[CIS0]	eTPU_C	eTPU Engine C Channel 0 Interrupt Status
0x1730	371	eTPU_GCR_CISR_ENG1[CIS1]	eTPU_C	eTPU Engine C Channel 1 Interrupt Status
0x1740	372	eTPU_GCR_CISR_ENG1[CIS2]	eTPU_C	eTPU Engine C Channel 2 Interrupt Status
0x1750	373	eTPU_GCR_CISR_ENG1[CIS3]	eTPU_C	eTPU Engine C Channel 3 Interrupt Status
0x1760	374	eTPU_GCR_CISR_ENG1[CIS4]	eTPU_C	eTPU Engine C Channel 4 Interrupt Status
0x1770	375	eTPU_GCR_CISR_ENG1[CIS5]	eTPU_C	eTPU Engine C Channel 5 Interrupt Status
0x1780	376	eTPU_GCR_CISR_ENG1[CIS6]	eTPU_C	eTPU Engine C Channel 6 Interrupt Status
0x1790	377	eTPU_GCR_CISR_ENG1[CIS7]	eTPU_C	eTPU Engine C Channel 7 Interrupt Status
0x17A0	378	eTPU_GCR_CISR_ENG1[CIS8]	eTPU_C	eTPU Engine C Channel 8 Interrupt Status
0x17B0	379	eTPU_GCR_CISR_ENG1[CIS9]	eTPU_C	eTPU Engine C Channel 9 Interrupt Status
0x17C0	380	eTPU_GCR_CISR_ENG1[CIS10]	eTPU_C	eTPU Engine C Channel 10 Interrupt Status
0x17D0	381	eTPU_GCR_CISR_ENG1[CIS11]	eTPU_C	eTPU Engine C Channel 11 Interrupt Status
0x17E0	382	eTPU_GCR_CISR_ENG1[CIS12]	eTPU_C	eTPU Engine C Channel 12 Interrupt Status

Table continues on the next page...

Table 4-4. Interrupt vectors and request sources (continued)

Offset	Vector	Source	Module	Description
0x17F0	383	eTPU_GCR_CISR_ENG1[CIS13]	eTPU_C	eTPU Engine C Channel 13 Interrupt Status
0x1800	384	eTPU_GCR_CISR_ENG1[CIS14]	eTPU_C	eTPU Engine C Channel 14 Interrupt Status
0x1810	385	eTPU_GCR_CISR_ENG1[CIS15]	eTPU_C	eTPU Engine C Channel 15 Interrupt Status
0x1820	386	eTPU_GCR_CISR_ENG1[CIS16]	eTPU_C	eTPU Engine C Channel 16 Interrupt Status
0x1830	387	eTPU_GCR_CISR_ENG1[CIS17]	eTPU_C	eTPU Engine C Channel 17 Interrupt Status
0x1840	388	eTPU_GCR_CISR_ENG1[CIS18]	eTPU_C	eTPU Engine C Channel 18 Interrupt Status
0x1850	389	eTPU_GCR_CISR_ENG1[CIS19]	eTPU_C	eTPU Engine C Channel 19 Interrupt Status
0x1860	390	eTPU_GCR_CISR_ENG1[CIS20]	eTPU_C	eTPU Engine C Channel 20 Interrupt Status
0x1870	391	eTPU_GCR_CISR_ENG1[CIS21]	eTPU_C	eTPU Engine C Channel 21 Interrupt Status
0x1880	392	eTPU_GCR_CISR_ENG1[CIS22]	eTPU_C	eTPU Engine C Channel 22 Interrupt Status
0x1890	393	eTPU_GCR_CISR_ENG1[CIS23]	eTPU_C	eTPU Engine C Channel 23 Interrupt Status
0x18A0	394	EQADCB_FISR _x [TORF] EQADCB_FISR _x [RFOF] EQADCB_FISR _x [CFUF]	eQADC_B	eQADC combined overrun interrupt requests from all of the FIFOs: Trigger Overrun, Receive FIFO Overflow, and command FIFO Underflow
0x18B0	395	EQADCB_FISR0[NCF]	eQADC_B	eQADC command FIFO 0 Non-Coherency Flag
0x18C0	396	EQADCB_FISR0[PF]	eQADC_B	eQADC command FIFO 0 Pause Flag
0x18D0	397	EQADCB_FISR0[EOQF]	eQADC_B	eQADC command FIFO 0 command queue End of Queue Flag
0x18E0	398	EQADCB_FISR0[CFFF]	eQADC_B	eQADC Command FIFO 0 Fill Flag
0x18F0	399	EQADCB_FISR0[RFDF]	eQADC_B	eQADC Receive FIFO 0 Drain Flag
0x1900	400	EQADCB_FISR1[NCF]	eQADC_B	eQADC command FIFO 1 Non-Coherency Flag
0x1910	401	EQADCB_FISR1[PF]	eQADC_B	eQADC command FIFO 1 Pause Flag
0x1920	402	EQADCB_FISR1[EOQF]	eQADC_B	eQADC command FIFO 1 command queue End of Queue Flag
0x1930	403	EQADCB_FISR1[CFFF]	eQADC_B	eQADC Command FIFO 1 Fill Flag
0x1940	404	EQADCB_FISR1[RFDF]	eQADC_B	eQADC Receive FIFO 1 Drain Flag
0x1950	405	EQADCB_FISR2[NCF]	eQADC_B	eQADC command FIFO 2 Non-Coherency Flag
0x1960	406	EQADCB_FISR2[PF]	eQADC_B	eQADC command FIFO 2 Pause Flag
0x1970	407	EQADCB_FISR2[EOQF]	eQADC_B	eQADC command FIFO 2 command queue End of Queue Flag
0x1980	408	EQADCB_FISR2[CFFF]	eQADC_B	eQADC Command FIFO 2 Fill Flag
0x1990	409	EQADCB_FISR2[RFDF]	eQADC_B	eQADC Receive FIFO 2 Drain Flag
0x19A0	410	EQADCB_FISR3[NCF]	eQADC_B	eQADC command FIFO 3 Non-Coherency Flag
0x19B0	411	EQADCB_FISR3[PF]	eQADC_B	eQADC command FIFO 3 Pause Flag
0x19C0	412	EQADCB_FISR3[EOQF]	eQADC_B	eQADC command FIFO 3 command queue End of Queue Flag
0x19D0	413	EQADCB_FISR3[CFFF]	eQADC_B	eQADC Command FIFO 3 Fill Flag

Table continues on the next page...

Table 4-4. Interrupt vectors and request sources (continued)

Offset	Vector	Source	Module	Description
0x19E0	414	EQADCB_FISR3[RFDF]	eQADC_B	eQADC Receive FIFO 3 Drain Flag
0x19F0	415	EQADCB_FISR4[NCF]	eQADC_B	eQADC command FIFO 4 Non-Coherency Flag
0x1A00	416	EQADCB_FISR4[PF]	eQADC_B	eQADC command FIFO 4 Pause Flag
0x1A10	417	EQADCB_FISR4[EOQF]	eQADC_B	eQADC command FIFO 4 command queue End of Queue Flag
0x1A20	418	EQADCB_FISR4[CFFF]	eQADC_B	eQADC Command FIFO 4 Fill Flag
0x1A30	419	EQADCB_FISR4[RFDF]	eQADC_B	eQADC Receive FIFO 4 Drain Flag
0x1A40	420	EQADCB_FISR5[NCF]	eQADC_B	eQADC command FIFO 5 Non-Coherency Flag
0x1A50	421	EQADCB_FISR5[PF]	eQADC_B	eQADC command FIFO 5 Pause Flag
0x1A60	422	EQADCB_FISR5[EOQF]	eQADC_B	eQADC command FIFO 5 command queue End of Queue Flag
0x1A70	423	EQADCB_FISR5[CFFF]	eQADC_B	eQADC Command FIFO 5 Fill Flag
0x1A80	424	EQADCB_FISR5[RFDF]	eQADC_B	eQADC Receive FIFO 5 Drain Flag
0x1A90	425	EDMAB_ERRH[ERR31:ERR0]	eDMA_B	eDMA channel Error flags 0-31
0x1AA0	426	EDMAB_IRQRH[INT0]	eDMA_B	eDMA channel Interrupt 0
0x1AB0	427	EDMAB_IRQRH[INT1]	eDMA_B	eDMA channel Interrupt 1
0x1AC0	428	EDMAB_IRQRH[INT2]	eDMA_B	eDMA channel Interrupt 2
0x1AD0	429	EDMAB_IRQRH[INT3]	eDMA_B	eDMA channel Interrupt 3
0x1AE0	430	EDMAB_IRQRH[INT4]	eDMA_B	eDMA channel Interrupt 4
0x1AF0	431	EDMAB_IRQRH[INT5]	eDMA_B	eDMA channel Interrupt 5
0x1B00	432	EDMAB_IRQRH[INT6]	eDMA_B	eDMA channel Interrupt 6
0x1B10	433	EDMAB_IRQRH[INT7]	eDMA_B	eDMA channel Interrupt 7
0x1B20	434	EDMAB_IRQRH[INT8]	eDMA_B	eDMA channel Interrupt 8
0x1B30	435	EDMAB_IRQRH[INT9]	eDMA_B	eDMA channel Interrupt 9
0x1B40	436	EDMAB_IRQRH[INT10]	eDMA_B	eDMA channel Interrupt 10
0x1B50	437	EDMAB_IRQRH[INT11]	eDMA_B	eDMA channel Interrupt 11
0x1B60	438	EDMAB_IRQRH[INT12]	eDMA_B	eDMA channel Interrupt 12
0x1B70	439	EDMAB_IRQRH[INT13]	eDMA_B	eDMA channel Interrupt 13
0x1B80	440	EDMAB_IRQRH[INT14]	eDMA_B	eDMA channel Interrupt 14
0x1B90	441	EDMAB_IRQRH[INT15]	eDMA_B	eDMA channel Interrupt 15
0x1BA0	442	EDMAB_IRQRH[INT16]	eDMA_B	eDMA channel Interrupt 16
0x1BB0	443	EDMAB_IRQRH[INT17]	eDMA_B	eDMA channel Interrupt 17
0x1BC0	444	EDMAB_IRQRH[INT18]	eDMA_B	eDMA channel Interrupt 18
0x1BD0	445	EDMAB_IRQRH[INT19]	eDMA_B	eDMA channel Interrupt 19
0x1BE0	446	EDMAB_IRQRH[INT20]	eDMA_B	eDMA channel Interrupt 20
0x1BF0	447	EDMAB_IRQRH[INT21]	eDMA_B	eDMA channel Interrupt 21
0x1C00	448	EDMAB_IRQRH[INT22]	eDMA_B	eDMA channel Interrupt 22
0x1C10	449	EDMAB_IRQRH[INT23]	eDMA_B	eDMA channel Interrupt 23
0x1C20	450	EDMAB_IRQRH[INT24]	eDMA_B	eDMA channel Interrupt 24

Table continues on the next page...

Table 4-4. Interrupt vectors and request sources (continued)

Offset	Vector	Source	Module	Description
0x1C30	451	EDMAB_IRQRH[INT25]	eDMA_B	eDMA channel Interrupt 25
0x1C40	452	EDMAB_IRQRH[INT26]	eDMA_B	eDMA channel Interrupt 26
0x1C50	453	EDMAB_IRQRH[INT27]	eDMA_B	eDMA channel Interrupt 27
0x1C60	454	EDMAB_IRQRH[INT28]	eDMA_B	eDMA channel Interrupt 28
0x1C70	455	EDMAB_IRQRH[INT29]	eDMA_B	eDMA channel Interrupt 29
0x1C80	456	EDMAB_IRQRH[INT30]	eDMA_B	eDMA channel Interrupt 30
0x1C90	457	EDMAB_IRQRH[INT31]	eDMA_B	eDMA channel Interrupt 31
0x1CA0	458	SD_ADC1 SD_ADC2 SD_ADC3 SD_ADC4	SDADC	SDADC1 to SDADC4 Interrupts
0x1CB0	459	eMIOS_1_GFLAG[F16]	eMIOS_1	eMIOS_1 channel 16 Flag
0x1CC0	460	eMIOS_1_GFLAG[F17]	eMIOS_1	eMIOS_1 channel 17 Flag
0x1CD0	461	eMIOS_1_GFLAG[F18]	eMIOS_1	eMIOS_1 channel 18 Flag
0x1CE0	462	eMIOS_1_GFLAG[F19]	eMIOS_1	eMIOS_1 channel 19 Flag
0x1CF0	463	eMIOS_1_GFLAG[F20]	eMIOS_1	eMIOS_1 channel 20 Flag
0x1D00	464	eMIOS_1_GFLAG[F21]	eMIOS_1	eMIOS_1 channel 21 Flag
0x1D10	465	eMIOS_1_GFLAG[F22]	eMIOS_1	eMIOS_1 channel 22 Flag
0x1D20	466	eMIOS_1_GFLAG[F23]	eMIOS_1	eMIOS_1 channel 23 Flag
0x1D30	467	DECFILTER_MSR_C[IDF]	DEC_C	Decimation C Input (Fill)
0x1D40	468	DECFILTER_MSR_C[ODFI SDF]	DEC_C	Decimation C Output/Integ (Drain/Integ)
0x1D50	469	DECFILTER C ERRORS	DEC_C	Decimation C Error
0x1D60	470	DECFILTER_MSR_D[IDF]	DEC_D	Decimation D Input (Fill)
0x1D70	471	DECFILTER_MSR_D[ODFI SDF]	DEC_D	Decimation D Output/Integ (Drain/Integ)
0x1D80	472	DECFILTER D ERRORS	DEC_D	Decimation D Error

Table continues on the next page...

Table 4-4. Interrupt vectors and request sources (continued)

Offset	Vector	Source	Module	Description
0x1D90	473	ESCIC_SR[TDRE] ESCIC_SR[TC] ESCIC_SR[RDRF] ESCIC_SR[IDLE] ESCIC_SR[OR] ESCIC_SR[NF] ESCIC_SR[FE] ESCIC_SR[PF] ESCIC_SR[BERR] ESCIC_SR[RXRDY] ESCIC_SR[TXRDY] ESCIC_SR[LWAKE] ESCIC_SR[STO] ESCIC_SR[PBERR] ESCIC_SR[CERR] ESCIC_SR[CKERR] ESCIC_SR[FRC] ESCIC_SR[OVFL]	eSCI_C	Combined Interrupt Requests of ESCI Module C: Transmit Data Register Empty, Transmit Complete, Receive Data Register Full, Idle line, Overrun, Noise Flag, Framing Error Flag, and Parity Error Flag interrupt requests, SCI Status Register 2 Bit Error interrupt request, LIN Status Register 1 Receive Data Ready, Transmit Data Ready, Received LIN Wakeup Signal, Slave TimeOut, Physical Bus Error, CRC Error, Checksum Error, Frame Complete interrupts requests, and LIN Status Register 2 Receive Register Overflow
0x1DA0	474	ESCID_SR[TDRE] ESCID_SR[TC] ESCID_SR[RDRF] ESCID_SR[IDLE] ESCID_SR[OR] ESCID_SR[NF] ESCID_SR[FE] ESCID_SR[PF] ESCID_SR[BERR] ESCID_SR[RXRDY] ESCID_SR[TXRDY] ESCID_SR[LWAKE] ESCID_SR[STO] ESCID_SR[PBERR] ESCID_SR[CERR] ESCID_SR[CKERR] ESCID_SR[FRC] ESCID_SR[OVFL]	eSCI_D	Combined Interrupt Requests of ESCI Module D: Transmit Data Register Empty, Transmit Complete, Receive Data Register Full, Idle line, Overrun, Noise Flag, Framing Error Flag, and Parity Error Flag interrupt requests, SCI Status Register 2 Bit Error interrupt request, LIN Status Register 1 Receive Data Ready, Transmit Data Ready, Received LIN Wakeup Signal, Slave TimeOut, Physical Bus Error, CRC Error, Checksum Error, Frame Complete interrupts requests, and LIN Status Register 2 Receive Register Overflow

Table continues on the next page...

Table 4-4. Interrupt vectors and request sources (continued)

Offset	Vector	Source	Module	Description
0x1DB0	475	ESCIE_SR[TDRE] ESCIE_SR[TC] ESCIE_SR[RDRF] ESCIE_SR[IDLE] ESCIE_SR[OR] ESCIE_SR[NF] ESCIE_SR[FE] ESCIE_SR[PF] ESCIE_SR[BERR] ESCIE_SR[RXRDY] ESCIE_SR[TXRDY] ESCIE_SR[LWAKE] ESCIE_SR[STO] ESCIE_SR[PBERR] ESCIE_SR[CERR] ESCIE_SR[CKERR] ESCIE_SR[FRC] ESCIE_SR[OVFL]	eSCI_E	Combined Interrupt Requests of ESCI Module E: Transmit Data Register Empty, Transmit Complete, Receive Data Register Full, Idle line, Overrun, Noise Flag, Framing Error Flag, and Parity Error Flag interrupt requests, SCI Status Register 2 Bit Error interrupt request, LIN Status Register 1 Receive Data Ready, Transmit Data Ready, Received LIN Wakeup Signal, Slave TimeOut, Physical Bus Error, CRC Error, Checksum Error, Frame Complete interrupts requests, and LIN Status Register 2 Receive Register Overflow
0x1DC0	476	DECFILTER_MSR_E[IDFIIODFII ERRORSISDF]	DEC_E	Decimation Filter E Input/Output/Err/Intg (Fill/ Drain/Error/Intg)
0x1DD0	477	DECFILTER_MSR_F[IDFIIODFII ERRORSISDF]	DEC_F	Decimation Filter F Input/Output/Err/Intg (Fill/ Drain/Error/Intg)
0x1DE0	478	DECFILTER_MSR_G[IDFIIODFII ERRORSISDF]	DEC_G	Decimation Filter G Input/Output/Err/Intg (Fill/ Drain/Error/Intg)
0x1DF0	479	DECFILTER_MSR_H[IDFIIODFII ERRORSISDF]	DEC_H	Decimation Filter H Input/Output/Err/Intg (Fill/ Drain/Error/Intg)
0x1E00	480	DECFILTER_MSR_I[IDFIIODFII ERRORSISDF]	DEC_I	Decimation Filter I Input/Output/Err/Intg (Fill/ Drain/Error/Intg)
0x1E10	481	DECFILTER_MSR_J[IDFIIODFII ERRORSISDF]	DEC_J	Decimation Filter J Input/Output/Err/Intg (Fill/ Drain/Error/Intg)
0x1E20	482	DECFILTER_MSR_K[IDFIIODFII ERRORSISDF]	DEC_K	Decimation Filter K Input/Output/Err/Intg (Fill/ Drain/Error/Intg)
0x1E30	483	DECFILTER_MSR_L[IDFIIODFII ERRORSISDF]	DEC_L	Decimation Filter L Input/Output/Err/Intg (Fill/ Drain/Error/Intg)
0x1E40	484	Reserved		Reserved for DEC_M
0x1E50	485	Reserved		Reserved for DEC_N
0x1E60	486	Reserved		Reserved for DEC_O
0x1E70	487	Reserved		Reserved for DEC_P
0x1E80	488	EDMAB_ERRH[ERR63:ERR32]	eDMA_B	eDMA channel Error flags 32-63
0x1E90	489	EDMAB_IRQRH[INT32:39]	eDMA_B	eDMA channel Interrupts 32-39
0x1EA0	490	EDMAB_IRQRH[INT40:47]	eDMA_B	eDMA channel Interrupts 40-47

Table continues on the next page...

Table 4-4. Interrupt vectors and request sources (continued)

Offset	Vector	Source	Module	Description
0x1EB0	491	EDMAB_IRQRH[INT48:55]	eDMA_B	eDMA channel Interrupts 48-55
0x1EC0	492	EDMAB_IRQRH[INT56:63]	eDMA_B	eDMA channel Interrupts 56-63
0x1ED0	493	eTPU_GCR_CISR_ENG1[CIS24]	eTPU_C	eTPU Engine C Channel 24 Interrupt Status
0x1EE0	494	eTPU_GCR_CISR_ENG1[CIS25]	eTPU_C	eTPU Engine C Channel 25 Interrupt Status
0x1EF0	495	eTPU_GCR_CISR_ENG1[CIS26]	eTPU_C	eTPU Engine C Channel 26 Interrupt Status
0x1F00	496	eTPU_GCR_CISR_ENG1[CIS27]	eTPU_C	eTPU Engine C Channel 27 Interrupt Status
0x1F10	497	eTPU_GCR_CISR_ENG1[CIS28]	eTPU_C	eTPU Engine C Channel 28 Interrupt Status
0x1F20	498	eTPU_GCR_CISR_ENG1[CIS29]	eTPU_C	eTPU Engine C Channel 29 Interrupt Status
0x1F30	499	eTPU_GCR_CISR_ENG1[CIS30]	eTPU_C	eTPU Engine C Channel 30 Interrupt Status
0x1F40	500	eTPU_GCR_CISR_ENG1[CIS31]	eTPU_C	eTPU Engine C Channel 31 Interrupt Status
0x1F50	501	SWTB_IR[TIF]	Watchdog B	Software Watchdog B Interrupt flag
0x1F60	502	SEMA4_CP0INE[0:15]	Semaphore	Core 0 requested semaphore has unlocked
0x1F70	503	SEMA4_CP1INE[0:15]	Semaphore	Core 1 requested semaphore has unlocked
0x1F80	504	CSE_IR[CIF]	CSE	CSE Interrupt
0x1F90	505	ESCIF_SR[TDRE] ESCIF_SR[TC] ESCIF_SR[RDRF] ESCIF_SR[IDLE] ESCIF_SR[OR] ESCIF_SR[NF] ESCIF_SR[FE] ESCIF_SR[PF] ESCIF_SR[BERR] ESCIF_SR[RXRDY] ESCIF_SR[TXRDY] ESCIF_SR[LWAKE] ESCIF_SR[STO] ESCIF_SR[PBERR] ESCIF_SR[CERR] ESCIF_SR[CKERR] ESCIF_SR[FRC] ESCIF_SR[OVFL]	eSCI_F	Combined Interrupt Requests of ESCI Module F: Transmit Data Register Empty, Transmit Complete, Receive Data Register Full, Idle line, Overrun, Noise Flag, Framing Error Flag, and Parity Error Flag interrupt requests, SCI Status Register 2 Bit Error interrupt request, LIN Status Register 1 Receive Data Ready, Transmit Data Ready, Received LIN Wakeup Signal, Slave TimeOut, Physical Bus Error, CRC Error, Checksum Error, Frame Complete interrupts requests, and LIN Status Register 2 Receive Register Overflow
0x1FA0	506	DSPIE_SR[TFUF] DSPIE_SR[RFOF] DSPIE_SR[SPEF] DSPIE_SR[DPEF]	DSPI_E	DSPI_E combined overrun and parity error interrupt requests: Transmit FIFO Underflow/Receive FIFO Overflow SPI Parity Error/DSI Parity Error
0x1FB0	507	DSPIE_SR[EOQF]	DSPI_E	DSPI_E transmit FIFO End of Queue Flag
0x1FC0	508	DSPIE_SR[TFFF]	DSPI_E	DSPI_E Transmit FIFO Fill Flag

Table continues on the next page...

Table 4-4. Interrupt vectors and request sources (continued)

Offset	Vector	Source	Module	Description
0x1FD0	509	DSPIE_SR[TCF] DSPIE_SR[DDIF]	DSPI_E	DSPI_E Transfer Complete/DSI Data Match Flag
0x1FE0	510	DSPIE_SR[RFDF]	DSPI_E	DSPI_E Receive FIFO Drain Flag
0x1FF0	511	STCU2_INT_FLG[MBIFLG]	STCU	MBIST interrupt

4.3 DMA request mapping

This chip has two separate eDMA controllers: eDMA_A and eDMA_B. Each has 64 channels.

The following tables show the sources of DMA requests for each controller.

Table 4-5. eDMA_A channel request allocation

Channel	Requesting source	Description
0	EQADC_A_FISR0[CFFF0]	EQADC_A Command FIFO 0 Fill Flag
1	EQADC_A_FISR0[RFDF0]	EQADC_A Receive FIFO 0 Drain Flag
2	EQADC_A_FISR1[CFFF1]	EQADC_A Command FIFO 1 Fill Flag
3	EQADC_A_FISR1[RFDF1]	EQADC_A Receive FIFO 1 Drain Flag
4	EQADC_A_FISR2[CFFF2]	EQADC_A Command FIFO 2 Fill Flag
5	EQADC_A_FISR2[RFDF2]	EQADC_A Receive FIFO 2 Drain Flag
6	EQADC_A_FISR3[CFFF3]	EQADC_A Command FIFO 3 Fill Flag
7	EQADC_A_FISR3[RFDF3]	EQADC_A Receive FIFO 3 Drain Flag
8	EQADC_A_FISR4[CFFF4]	EQADC_A Command FIFO 4 Fill Flag
9	EQADC_A_FISR4[RFDF4]	EQADC_A Receive FIFO 4 Drain Flag
10	EQADC_A_FISR5[CFFF5]	EQADC_A Command FIFO 5 Fill Flag
11	EQADC_A_FISR5[RFDF5]	EQADC_A Receive FIFO 5 Drain Flag
12	DSPI_B_SR[TFFF]	DSPI_B Transmit FIFO Fill Flag
13	DSPI_B_SR[RFDF]	DSPI_B Receive FIFO Drain Flag
14	DSPI_C_SR[TFFF]	DSPI_C Transmit FIFO Fill Flag
15	DSPI_C_SR[RFDF]	DSPI_C Receive FIFO Drain Flag
16	DSPI_D_SR[TFFF]	DSPI_D Transmit FIFO Fill Flag
17	DSPI_D_SR[RFDF]	DSPI_D Receive FIFO Drain Flag
18	eSCI_A_IFSR1[TDRE] eSCI_A_IFSR1[TC] eSCI_A_IFSR2[TXRDY]	eSCI_A combined DMA request of the Transmit Data Register Empty, Transmit Complete, and LIN Transmit Data Ready DMA requests
19	eSCI_A_IFSR1[RDRF] eSCI_A_IFSR2[RXRDY]	eSCI_A combined DMA request of the Receive Data Register Full and LIN Receive Data Ready DMA requests
20	eMIOS_0_GFLAG[F0]	eMIOS_0 channel 0 Flag

Table continues on the next page...

Table 4-5. eDMA_A channel request allocation (continued)

Channel	Requesting source	Description
21	eMIOS_0_GFLAG[F1]	eMIOS_0 channel 1 Flag
22	eMIOS_0_GFLAG[F2]	eMIOS_0 channel 2 Flag
23	eMIOS_0_GFLAG[F3]	eMIOS_0 channel 3 Flag
24	eMIOS_0_GFLAG[F4]	eMIOS_0 channel 4 Flag
25	eMIOS_1_GFLAG[F0]	eMIOS_1 channel 0 Flag
26	eMIOS_1_GFLAG[F1]	eMIOS_1 channel 1 Flag
27	eTPU_AB_GCR_CDTRSR_ENG1[DTRS0]	eTPU_A channel 0 Data Transfer Request Status
28	eTPU_AB_GCR_CDTRSR_ENG1[DTRS1]	eTPU_A channel 1 Data Transfer Request Status
29	eTPU_AB_GCR_CDTRSR_ENG1[DTRS2]	eTPU_A channel 2 Data Transfer Request Status
30	eTPU_AB_GCR_CDTRSR_ENG1[DTRS14]	eTPU_A channel 14 Data Transfer Request Status
31	eTPU_AB_GCR_CDTRSR_ENG1[DTRS15]	eTPU_A channel 15 Data Transfer Request Status
32	DSPI_A_SR[TFFF]	DSPI_A Transmit FIFO Fill Flag
33	DSPI_A_SR[RDFD]	DSPI_A Receive FIFO Drain Flag
34	eSCI_B_IFSR1[TDRE] eSCI_B_IFSR1[TC] eSCI_B_IFSR2[TXRDY]	eSCI_B combined DMA request of the Transmit Data Register Empty, Transmit Complete, and LIN Transmit Data Ready DMA requests
35	eSCI_B_IFSR1[RDRF] eSCI_B_IFSR2[RXRDY]	eSCI_B combined DMA request of the Receive Data Register Full and LIN Receive Data Ready DMA requests
36	eMIOS_0_GFLAG[F6]	eMIOS_0 channel 6 Flag
37	eMIOS_0_GFLAG[F7]	eMIOS_0 channel 7 Flag
38	eMIOS_1_GFLAG[F2]	eMIOS_1 channel 2 Flag
39	eMIOS_1_GFLAG[F3]	eMIOS_1 channel 3 Flag
40	eMIOS_0_GFLAG[F16]	eMIOS_0 channel 16 Flag
41	eMIOS_0_GFLAG[F17]	eMIOS_0 channel 17 Flag
42	eMIOS_0_GFLAG[F18]	eMIOS_0 channel 18 Flag
43	eMIOS_0_GFLAG[F19]	eMIOS_0 channel 19 Flag
44	eTPU_AB_GCR_CDTRSR_ENG1[DTRS12]	eTPU_A channel 12 Data Transfer Request Status
45	eTPU_AB_GCR_CDTRSR_ENG1[DTRS13]	eTPU_A channel 13 Data Transfer Request Status
46	eTPU_AB_GCR_CDTRSR_ENG1[DTRS28]	eTPU_A channel 28 Data Transfer Request Status
47	eTPU_AB_GCR_CDTRSR_ENG1[DTRS29]	eTPU_A channel 29 Data Transfer Request Status
48	SIU_EISR[EIF0]	SIU External Interrupt Flag 0
49	SIU_EISR[EIF1]	SIU External Interrupt Flag 1
50	SIU_EISR[EIF2]	SIU External Interrupt Flag 2
51	SIU_EISR[EIF3]	SIU External Interrupt Flag 3
52	eTPU_AB_GCR_CDTRSR_ENG2[DTRS0]	eTPU_B channel 0 Data Transfer Request Status
53	eTPU_AB_GCR_CDTRSR_ENG2[DTRS1]	eTPU_B channel 1 Data Transfer Request Status
54	eTPU_AB_GCR_CDTRSR_ENG2[DTRS2]	eTPU_B channel 2 Data Transfer Request Status
55	eTPU_AB_GCR_CDTRSR_ENG2[DTRS3]	eTPU_B channel 3 Data Transfer Request Status
56	eTPU_AB_GCR_CDTRSR_ENG2[DTRS12]	eTPU_B channel 12 Data Transfer Request Status
57	eTPU_AB_GCR_CDTRSR_ENG2[DTRS13]	eTPU_B channel 13 Data Transfer Request Status

Table continues on the next page...

Table 4-5. eDMA_A channel request allocation (continued)

Channel	Requesting source	Description
58	eTPU_AB_GCR_CDTRSR_ENG2[DTRS14]	eTPU_B channel 14 Data Transfer Request Status
59	eTPU_AB_GCR_CDTRSR_ENG2[DTRS15]	eTPU_B channel 15 Data Transfer Request Status
60	eTPU_AB_GCR_CDTRSR_ENG2[DTRS28]	eTPU_B channel 28 Data Transfer Request Status
61	eTPU_AB_GCR_CDTRSR_ENG2[DTRS29]	eTPU_B channel 29 Data Transfer Request Status
62	eTPU_AB_GCR_CDTRSR_ENG2[DTRS30]	eTPU_B channel 30 Data Transfer Request Status
63	eTPU_AB_GCR_CDTRSR_ENG2[DTRS31]	eTPU_B channel 31 Data Transfer Request Status

Table 4-6. eDMA_B channel request allocation

Channel	Requesting source	Description
0	EQADC_B_FISR0[CFFF0]	EQADC_B Command FIFO 0 Fill Flag
1	EQADC_B_FISR0[RFDF0]	EQADC_B Result FIFO 0 Drain Flag
2	EQADC_B_FISR1[CFFF1]	EQADC_B Command FIFO 1 Fill Flag
3	EQADC_B_FISR1[RFDF1]	EQADC_B Result FIFO 1 Drain Flag
4	EQADC_B_FISR2[CFFF2]	EQADC_B Command FIFO 2 Fill Flag
5	EQADC_B_FISR2[RFDF2]	EQADC_B Result FIFO 2 Drain Flag
6	EQADC_B_FISR3[CFFF3]	EQADC_B Command FIFO 3 Fill Flag
7	EQADC_B_FISR3[RFDF3]	EQADC_B Result FIFO 3 Drain Flag
8	EQADC_B_FISR4[CFFF4]	EQADC_B Command FIFO 4 Fill Flag
9	EQADC_B_FISR4[RFDF4]	EQADC_B Result FIFO 4 Drain Flag
10	EQADC_B_FISR5[CFFF5]	EQADC_B Command FIFO 5 Fill Flag
11	EQADC_B_FISR5[RFDF5]	EQADC_B Result FIFO 5 Drain Flag
12	DECFILTERA_IB	Decimation Filter A Input Buffer (fill)
13	DECFILTERA_OB_INTEG	Decimation Filter A Output Buffer/Integrator (drain/integrator)
14	DECFILTERB_IB	Decimation Filter B Input Buffer (fill)
15	DECFILTERB_OB_INTEG	Decimation Filter B Output Buffer/Integrator (drain/integrator)
16	DECFILTERC_IB	Decimation Filter C Input Buffer (fill)
17	DECFILTERC_OB_INTEG	Decimation Filter C Output Buffer/Integrator (drain/integrator)
18	DECFILTERD_IB	Decimation Filter D Input Buffer (fill)
19	DECFILTERD_OB_INTEG	Decimation Filter D Output Buffer/Integrator (drain/integrator)
20	DECFILTERE_IB	Decimation Filter E Input Buffer (fill)
21	DECFILTERE_OB_INTEG	Decimation Filter E Output Buffer/Integrator (drain/integrator)
22	DECFILTERF_IB	Decimation Filter F Input Buffer (fill)
23	DECFILTERF_OB_INTEG	Decimation Filter F Output Buffer/Integrator (drain/integrator)
24	DECFILTERG_IB	Decimation Filter G Input Buffer (fill)
25	DECFILTERG_OB_INTEG	Decimation Filter G Output Buffer/Integrator (drain/integrator)
26	DECFILTERH_IB	Decimation Filter H Input Buffer (fill)
27	DECFILTERH_OB_INTEG	Decimation Filter H Output Buffer/Integrator (drain/integrator)
28	DECFILTERI_IB	Decimation Filter I Input Buffer (fill)

Table continues on the next page...

Table 4-6. eDMA_B channel request allocation (continued)

Channel	Requesting source	Description
29	DECFILTERI_OB_INTEG	Decimation Filter I Output Buffer/Integrator (drain/integrator)
30	DECFILTERJ_IB	Decimation Filter J Input Buffer (fill)
31	DECFILTERJ_OB_INTEG	Decimation Filter J Output Buffer/Integrator (drain/integrator)
32	DECFILTERK_IB	Decimation Filter K Input Buffer (fill)
33	DECFILTERK_OB_INTEG	Decimation Filter K Output Buffer/Integrator (drain/integrator)
34	DECFILTERL_IB	Decimation Filter L Input Buffer (fill)
35	DECFILTERL_OB_INTEG	Decimation Filter L Output Buffer/Integrator (drain/integrator)
36	SDADC_0	SDADC_0 result ready
37	SDADC_1	SDADC_1 result ready
38	SDADC_2	SDADC_2 result ready
39	SDADC_3	SDADC_3 result ready
40	SRX_0 Fast Message Ready	SENT Receiver Module 0 Fast Message Ready
41	SRX_1 Fast Message Ready	SENT Receiver Module 1 Fast Message Ready
42	SRX_0 Slow Serial Message Ready	SENT Receiver Module 0 Slow Serial Message Ready
43	SRX_1 Slow Serial Message Ready	SENT Receiver Module 1 Slow Serial Message Ready
44	eTPU_C_GCR_CDTRSR_ENG1[DTRS0]	eTPU_C Channel 0 Data Transfer Request Status
45	eTPU_C_GCR_CDTRSR_ENG1[DTRS1]	eTPU_C Channel 1 Data Transfer Request Status
46	eTPU_C_GCR_CDTRSR_ENG1[DTRS2]	eTPU_C Channel 2 Data Transfer Request Status
47	eTPU_C_GCR_CDTRSR_ENG1[DTRS3]	eTPU_C Channel 3 Data Transfer Request Status
48	eTPU_C_GCR_CDTRSR_ENG1[DTRS4]	eTPU_C Channel 4 Data Transfer Request Status
49	eTPU_C_GCR_CDTRSR_ENG1[DTRS5]	eTPU_C Channel 5 Data Transfer Request Status
50	eTPU_C_GCR_CDTRSR_ENG1[DTRS6]	eTPU_C Channel 6 Data Transfer Request Status
51	eTPU_C_GCR_CDTRSR_ENG1[DTRS7]	eTPU_C Channel 7 Data Transfer Request Status
52	PSI5_0 SMC	PSI5_0 DMA Request for SMC
53	PSI5_0 DIAG	PSI5_0 DMA Request for Diagnostics
54	PSI5_1 SMC	PSI5_1 DMA Request for SMC
55	PSI5_1 DIAG	PSI5_1 DMA Request for Diagnostics
56	SIPI CH0	SIPI Channel 0 DMA Request
57	SIPI CH1	SIPI Channel 1 DMA Request
58	SIPI CH2	SIPI Channel 2 DMA Request
59	SIPI CH3	SIPI Channel 3 DMA Request
60	M_CAN_0	M_CAN_0 DMA Request
61	DSPI_E_SR[TFFF]	DSPI_E Transmit FIFO Fill Flag
62	DSPI_E_SR[RDFD]	DSPI_E Receive FIFO Drain Flag
63	M_CAN_1	M_CAN_1 DMA Request

Chapter 5

Device Configuration Format (DCF) Records

5.1 Overview

Device Configuration Format (DCF) records are used by the System Status and Configuration Module (SSCM) to configure certain registers in the device during system boot while the reset signal is asserted.

There are two kinds of DCF records:

- Factory-written DCF records: written to UTEST flash memory by the factory.
- User-supplied DCF records: written to UTEST flash memory by the user.

The following figure is a high-level conceptual representation of the functionality of DCF records.

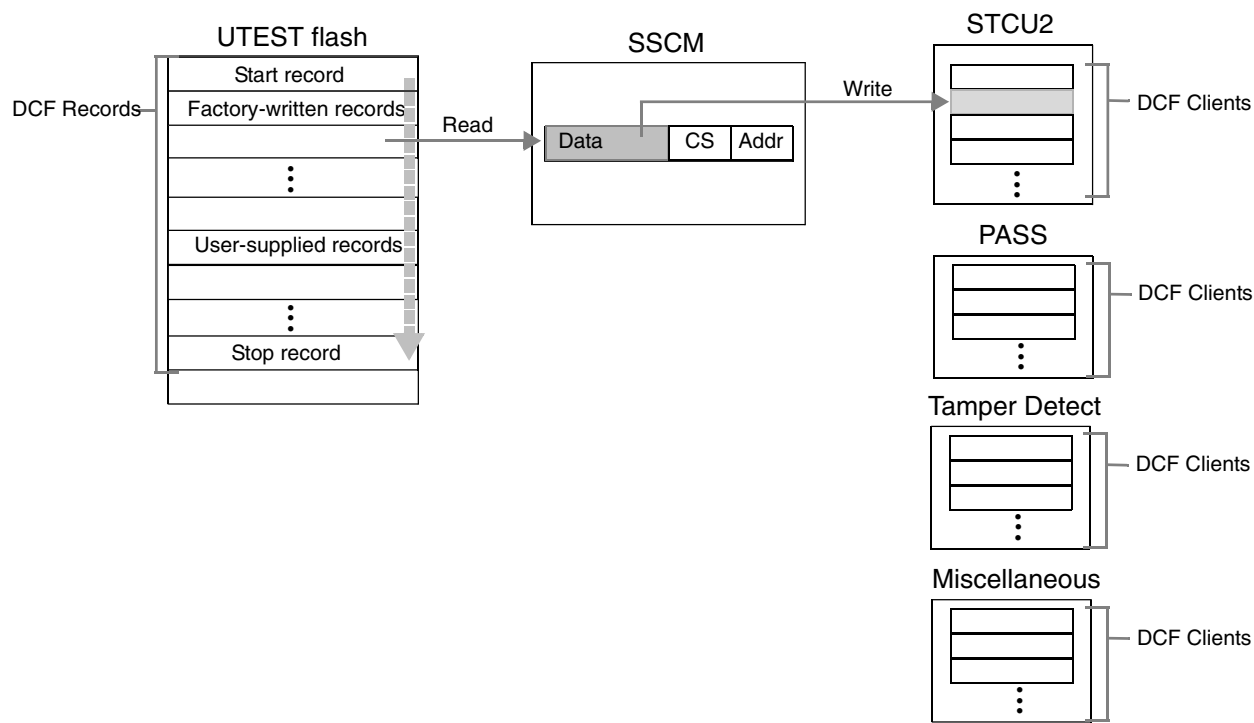


Figure 5-1. Conceptual representation of DCF record functionality

5.2 DCF records in boot sequence

The following is a high-level summary of the boot sequence leading to the use of DCF records:

1. When power is applied to a properly programmed device, the Power Management Controller (PMC) takes control of the device.
2. After the system power supplies have reached predefined levels, the flash memory array is initialized and the SSCM is enabled.
3. The SSCM reads the DCF records.
4. The SSCM writes the configuration information to the specified registers.
5. After registers are initialized: If the STCU2 is enabled by DCF record information, it begins executing the tasks enabled by its DCF record configuration.

5.3 DCF records location

Factory-written DCF records start at the first address allocated to the DCF records in the UTEST flash memory area. See the UTEST flash memory map.

User-supplied UTEST DCF records may be added at the next location in the UTEST memory map immediately following the factory-written UTEST DCF records.

5.4 DCF record structure

A DCF record is a contiguous double-word (64-bit) entry consisting of the following:

- Control word: Information to locate the corresponding DCF client internal to the device (pointer to the location of a register internal to the device)
- Data word: Data to be written to that client

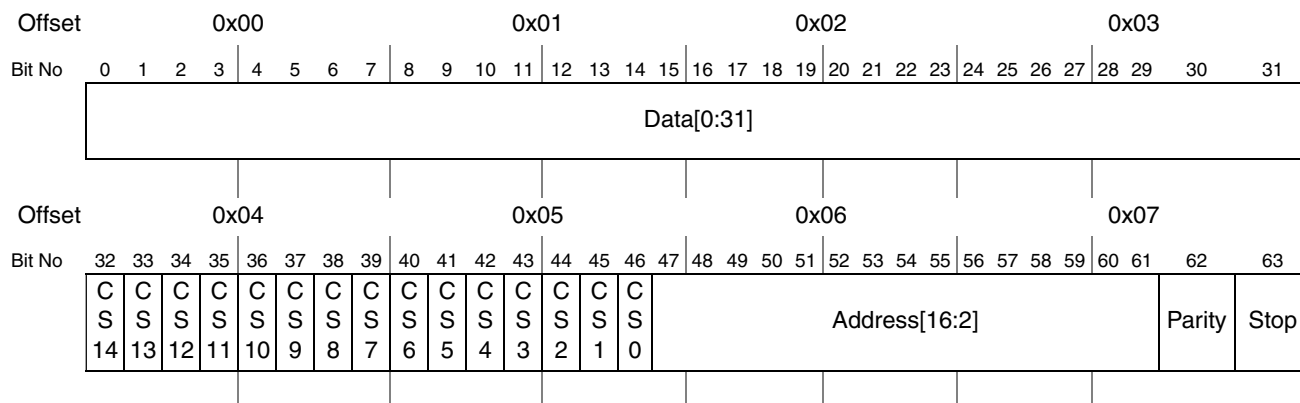


Figure 5-2. DCF record structure

Table 5-1. DCF record field descriptions

Field	Name	Description
0-31	Data[0:31]	32 bits of data to be written to the DCF client
32-46	CS _n	Chip Select <i>n</i> For each DCF record, one Chip Select bit is asserted (1b) to select the target module for the DCF client. All other Chip Select bits per DCF record should be negated (0b).
47-61	Address[16:2]	Address of the DCF client within the selected module NOTE: Address decoding for DCF clients may not match the standard software address map decoding. Details of DCF client addresses are defined in each applicable module chapter.
62	Parity	Parity bit for the DCF record
63	Stop	Stop bit Indicates the end of the list of DCF records. 0 indicates the DCF record is not the end of the list 1 indicates the DCF record is the end of the list NOTE: The erased state of flash memory is 0xFFFF_FFFF_FFFF_FFFF. As a result, the list ends with the first unprogrammed double word. This location can be programmed with a new record to extend the list.

5.5 DCF records sequence

DCF records must appear as contiguous series of entries programmed from the beginning of the UTEST flash memory area in the following order.

1. **Start DCF record:** The first DCF record must be a start record. This record must be found at the beginning of a DCF area in flash memory to indicate to the device that the following records must be processed.
2. **DCF records containing configuration data:** DCF records containing configuration data must immediately follow the start record with no blank records in between.
3. **Stop DCF record:** The end of the configuration records is indicated by the stop record. A record with the stop bit set is a stop record. All other bits are ignored. No DCF records following the stop record are processed.

SSCM recognizes the following record as a start record.

Table 5-2. DCF start record

0x00 (0:31)	0x04 (32:63)
0x05AA_55AF	0x0000_0000

The start bit is set by the factory at the beginning of the UTEST flash memory area.

SSCM recognizes the following record as a stop record.

Table 5-3. DCF stop record

0:31	32:62	63
Ignored	Ignored	1

The stop bit is never set by the factory, since this would not allow the user to add DCF records. The flash memory location following the last factory-written UTEST DCF record is unprogrammed, with the content of 0xFFFF_FFFF. This is interpreted as the stop record, since the stop bit is set.

User-supplied DCF records may be added in a contiguous manner immediately following the factory-written DCF records. There must never be an unprogrammed record in the series of DCF records as it will be interpreted as a stop record.

The following table shows the series of DCF records when n data records are stored in the UTEST flash.

Table 5-4. Series of DCF records in UTEST flash memory

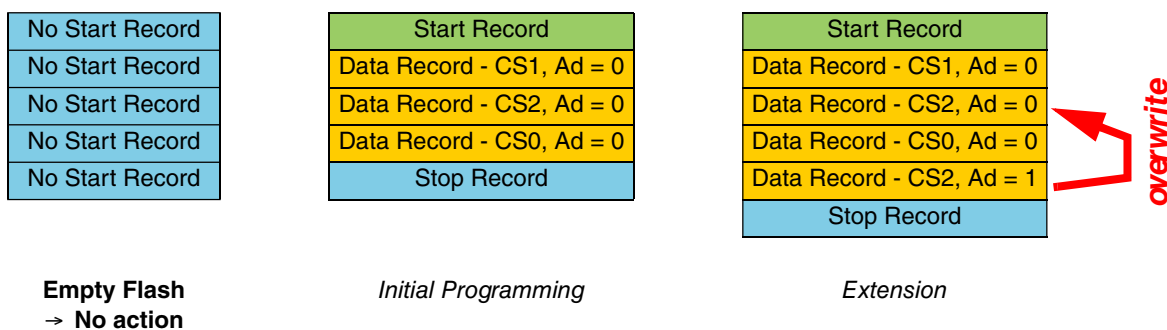
Record type	ADDR offset	DATA	
Start record	0x00	0x05AA_55AF	
	0x04	0x0000_0000	STOP=0

Table continues on the next page...

Table 5-4. Series of DCF records in UTEST flash memory (continued)

Record type	ADDR offset	DATA			
Data records	0x08	WDATA[31:0]			
	0x0C	CS[14:0]	ADDR[16:2]	PRTY	STOP=0
	0x10	WDATA[31:0]			
	0x14	CS[14:0]	ADDR[16:2]	PRTY	STOP=0
	...	...			
Stop record	$8n-1 + 0x0$	Reserved			
	$8n-1 + 0x4$	Reserved			1
	$8n + 0x0$				
	$8n + 0x4$				

It is possible for more than one DCF Record to write to the same DCF client. In this case the later record usually overrides a DCF client value set by a previous record. However, not all DCF clients allow overwrites; this depends on the DCF client implementation.

**Figure 5-3. Appending DCF records**

5.6 DCF client Chip Select module assignments

The following table shows the Chip Select (CS) for each group of DCF clients. A CS can be assigned to one module or multiple modules, as indicated. CSs not listed are reserved.

Table 5-5. DCF client CS module assignments

CS	DCF client target module(s)
CS[0]	Reserved
CS[1]	SSCM (Reserved)
CS[2]	STCU2
CS[3]	PASS

Table continues on the next page...

Table 5-5. DCF client CS module assignments (continued)

CS	DCF client target module(s)
CS[4]	Tamper Detect (TDM)
CS[5]	Reserved
CS[6]	Reserved
CS[7]	Miscellaneous CSE Secure Boot Configurations Analog Oscillator Configurations ERROR Functions Pads Configuration Reset Escalation PMC Trimming

5.7 DCF clients

DCF clients are 32-bit-wide hardware registers inside a module that receive and store the data from a DCF record. This stored data is used to initialize registers and to configure features. Their characteristics include:

- DCF clients have a default value before any DCF records are written.
- DCF clients are not required to implement all 32 bits.

The following table below shows the DCF clients that are available in this chip's UTEST area of flash memory. The following list of terms defines the meaning of the table's column headings.

DCF CS: Chip Select

The CS field is a 15-bit field with each bit in the field assigned to a particular module. These bits act as module select signals for the internal modules. Only one bit in this field should be set in a DCF record.

DCF Address[10:2]

The Address[16:2] field of a DCF record specifies an internal address for a DCF client. The following table provides the Address[10:2] portion of the field. The address is used only by the SSCM when interpreting DCF records and writing the data to specified modules.

DCF client description

This entry contains the name of the DCF client.

DCF client reset value

This field contains one of the following:

- the reset value of the DCF client before the SSCM writes the client with the associated DCF record
- a factory-programmed value

DCF client target module

This entry identifies the module that uses the DCF client information.

DCF client special strategy

Special requirements apply to how some DCF clients are written. The SSCM examines the DCF client to determine any correct special strategy to use and then appropriately writes a DCF record to that client.

None

No special DCF strategy is used.

Parity

On this chip, parity is implemented only for PMC DCF clients.

Triple Voted

DCF clients that use the Triple Voted strategy have three copies of the register. The SSCM writes to all three registers in a single write cycle. The outputs of the three registers are majority voted together to determine the correct data value. Triple voting allows a bit-flip error to occur without changing the DCF client output data.

write_once

A register using the Write Once strategy can only be written once. The DCF client ignores subsequent writes.

CAUTION

Plan carefully before writing write-once DCF clients. Any DCF client that has a write-once attribute cannot be reprogrammed by user software until the next reset or by more than one DCF record. Subsequent DCF records are ignored.

wr0_only: Write 0 only

A bit in a DCF client can be written only from a logic 1 to a logic 0. An attempt to write a bit with this attribute to a logic 1 is ignored.

wr1_only: Write 1 only

A bit in a DCF client can be written only from a logic 0 to a logic 1. An attempt to write a bit with this attribute to a logic 0 is ignored.

Any UTEST DCF client can be programmed as UTEST non-differential.

Table 5-6. UTEST DCF clients

DCF CS[14:0]	DCF Address[10:2]	DCF client description	DCF client reset value (hex)	DCF client target module	DCF client special strategy
STCU2: support for 6 LBIST partitions; 64-bit MISR					
000_0000_0000_0100	0_0000_0010	STCU_SKC	0000_0000	STCU	None
000_0000_0000_0100	0_0000_0000	STCU_RUN	0000_0000	STCU	None
000_0000_0000_0100	0_0000_0011	STCU_CFG	0000_0000	STCU	None
000_0000_0000_0100	0_0000_0100	STCU_PLL_CFG	0000_0000	STCU	None
000_0000_0000_0100	0_0000_0101	STCU_WDG	0000_FFFF	STCU	None
000_0000_0000_0100	0_0000_0111	Reserved			
000_0000_0000_0100	0_0000_1010	STCU_ERR_FM	0000_0000	STCU	None
000_0000_0000_0100	0_0000_1111	STCU_LBRMSW	0000_003F	STCU	None
000_0000_0000_0100	0_0001_0000	STCU_LBUFM	0000_0000	STCU	None
000_0000_0000_0100	0_0001_1101	STCU_MBUFML	0000_0000	STCU	None
000_0000_0000_0100	0_0001_1110	STCU_MBUFMM	0000_0000	STCU	None
000_0000_0000_0100	0_0001_1111	STCU_MBUFMH	0000_0000	STCU	None
000_0000_0000_0100	0_0100_0000	STCU_LB_CTRL_0	0000_0000	STCU	None
000_0000_0000_0100	0_0100_0001	STCU_LB_PCS_0	0000_0000	STCU	None
000_0000_0000_0100	0_0100_0010	Reserved			
000_0000_0000_0100	0_0100_0011	Reserved			
000_0000_0000_0100	0_0100_0100	STCU_LB_MISREL_0	FFFF_FFFF	STCU	None
000_0000_0000_0100	0_0100_0101	STCU_LB_MISREH_0	FFFF_FFFF	STCU	None
000_0000_0000_0100	0_0100_1000	STCU_LB_MISRELSW_0	FFFF_FFFF	STCU	None
000_0000_0000_0100	0_0100_1001	STCU_LB_MISREHSW_0	FFFF_FFFF	STCU	None
000_0000_0000_0100	0_0101_0000	STCU_LB_CTRL_1	0000_0000	STCU	None
000_0000_0000_0100	0_0101_0001	STCU_LB_PCS_1	0000_0000	STCU	None
000_0000_0000_0100	0_0101_0010	Reserved			
000_0000_0000_0100	0_0101_0011	Reserved			
000_0000_0000_0100	0_0101_0100	STCU_LB_MISREL_1	FFFF_FFFF	STCU	None
000_0000_0000_0100	0_0101_0101	STCU_LB_MISREH_1	FFFF_FFFF	STCU	None
000_0000_0000_0100	0_0101_1000	STCU_LB_MISRELSW_1	FFFF_FFFF	STCU	None
000_0000_0000_0100	0_0101_1001	STCU_LB_MISREHSW_1	FFFF_FFFF	STCU	None
000_0000_0000_0100	0_0110_0000	STCU_LB_CTRL_2	0000_0000	STCU	None
000_0000_0000_0100	0_0110_0001	STCU_LB_PCS_2	0000_0000	STCU	None
000_0000_0000_0100	0_0110_0010	Reserved			
000_0000_0000_0100	0_0110_0011	Reserved			
000_0000_0000_0100	0_0110_0100	STCU_LB_MISREL_2	FFFF_FFFF	STCU	None

Table continues on the next page...

Table 5-6. UTEST DCF clients (continued)

DCF CS[14:0]	DCF Address[10:2]	DCF client description	DCF client reset value (hex)	DCF client target module	DCF client special strategy
000_0000_0000_0100	0_0110_0101	STCU_LB_MISREH_2	FFFF_FFFF	STCU	None
000_0000_0000_0100	0_0110_1000	STCU_LB_MISRELSW_2	FFFF_FFFF	STCU	None
000_0000_0000_0100	0_0110_1001	STCU_LB_MISREHSW_2	FFFF_FFFF	STCU	None
000_0000_0000_0100	0_0111_0000	STCU_LB_CTRL_3	0000_0000	STCU	None
000_0000_0000_0100	0_0111_0001	STCU_LB_PCS_3	0000_0000	STCU	None
000_0000_0000_0100	0_0111_0010	Reserved			
000_0000_0000_0100	0_0111_0011	Reserved			
000_0000_0000_0100	0_0111_0100	STCU_LB_MISREL_3	FFFF_FFFF	STCU	None
000_0000_0000_0100	0_0111_0101	STCU_LB_MISREH_3	FFFF_FFFF	STCU	None
000_0000_0000_0100	0_0111_1000	STCU_LB_MISRELSW_3	FFFF_FFFF	STCU	None
000_0000_0000_0100	0_0111_1001	STCU_LB_MISREHSW_3	FFFF_FFFF	STCU	None
000_0000_0000_0100	0_1000_0000	STCU_LB_CTRL_4	0000_0000	STCU	None
000_0000_0000_0100	0_1000_0001	STCU_LB_PCS_4	0000_0000	STCU	None
000_0000_0000_0100	0_1000_0010	Reserved			
000_0000_0000_0100	0_1000_0011	Reserved			
000_0000_0000_0100	0_1000_0100	STCU_LB_MISREL_4	FFFF_FFFF	STCU	None
000_0000_0000_0100	0_1000_0101	STCU_LB_MISREH_4	FFFF_FFFF	STCU	None
000_0000_0000_0100	0_1000_1000	STCU_LB_MISRELSW_4	FFFF_FFFF	STCU	None
000_0000_0000_0100	0_1000_1001	STCU_LB_MISREHSW_4	FFFF_FFFF	STCU	None
000_0000_0000_0100	0_1001_0000	STCU_LB_CTRL_5	0000_0000	STCU	None
000_0000_0000_0100	0_1001_0001	STCU_LB_PCS_5	0000_0000	STCU	None
000_0000_0000_0100	0_1001_0010	Reserved			
000_0000_0000_0100	0_1001_0011	Reserved			
000_0000_0000_0100	0_1001_0100	STCU_LB_MISREL_5	FFFF_FFFF	STCU	None
000_0000_0000_0100	0_1001_0101	STCU_LB_MISREH_5	FFFF_FFFF	STCU	None
000_0000_0000_0100	0_1001_1000	STCU_LB_MISRELSW_5	FFFF_FFFF	STCU	None
000_0000_0000_0100	0_1001_1001	STCU_LB_MISREHSW_5	FFFF_FFFF	STCU	None
000_0000_0000_0100	1_1000_0000	STCU_MB_CTRL_0	0000_0000	STCU	None
000_0000_0000_0100	1_1000_0001	STCU_MB_CTRL_1	0000_0000	STCU	None
000_0000_0000_0100	1_1000_0010	STCU_MB_CTRL_2	0000_0000	STCU	None
000_0000_0000_0100	1_1000_0011	STCU_MB_CTRL_3	0000_0000	STCU	None
000_0000_0000_0100	1_1000_0100	STCU_MB_CTRL_4	0000_0000	STCU	None
000_0000_0000_0100	1_1000_0101	STCU_MB_CTRL_5	0000_0000	STCU	None
000_0000_0000_0100	1_1000_0110	STCU_MB_CTRL_6	0000_0000	STCU	None
000_0000_0000_0100	1_1000_0111	STCU_MB_CTRL_7	0000_0000	STCU	None
000_0000_0000_0100	1_1000_1000	STCU_MB_CTRL_8	0000_0000	STCU	None
000_0000_0000_0100	1_1000_1001	STCU_MB_CTRL_9	0000_0000	STCU	None
000_0000_0000_0100	1_1000_1010	STCU_MB_CTRL_10	0000_0000	STCU	None
000_0000_0000_0100	1_1000_1011	STCU_MB_CTRL_11	0000_0000	STCU	None

Table continues on the next page...

Table 5-6. UTEST DCF clients (continued)

DCF CS[14:0]	DCF Address[10:2]	DCF client description	DCF client reset value (hex)	DCF client target module	DCF client special strategy
000_0000_0000_0100	1_1000_1100	STCU_MB_CTRL_12	0000_0000	STCU	None
000_0000_0000_0100	1_1000_1101	STCU_MB_CTRL_13	0000_0000	STCU	None
000_0000_0000_0100	1_1000_1110	STCU_MB_CTRL_14	0000_0000	STCU	None
000_0000_0000_0100	1_1000_1111	STCU_MB_CTRL_15	0000_0000	STCU	None
000_0000_0000_0100	1_1001_0000	STCU_MB_CTRL_16	0000_0000	STCU	None
000_0000_0000_0100	1_1001_0001	STCU_MB_CTRL_17	0000_0000	STCU	None
000_0000_0000_0100	1_1001_0010	STCU_MB_CTRL_18	0000_0000	STCU	None
000_0000_0000_0100	1_1001_0011	STCU_MB_CTRL_19	0000_0000	STCU	None
000_0000_0000_0100	1_1001_0100	STCU_MB_CTRL_20	0000_0000	STCU	None
000_0000_0000_0100	1_1001_0101	STCU_MB_CTRL_21	0000_0000	STCU	None
000_0000_0000_0100	1_1001_0110	STCU_MB_CTRL_22	0000_0000	STCU	None
000_0000_0000_0100	1_1001_0111	STCU_MB_CTRL_23	0000_0000	STCU	None
000_0000_0000_0100	1_1001_1000	STCU_MB_CTRL_24	0000_0000	STCU	None
000_0000_0000_0100	1_1001_1001	STCU_MB_CTRL_25	0000_0000	STCU	None
000_0000_0000_0100	1_1001_1010	STCU_MB_CTRL_26	0000_0000	STCU	None
000_0000_0000_0100	1_1001_1011	STCU_MB_CTRL_27	0000_0000	STCU	None
000_0000_0000_0100	1_1001_1100	STCU_MB_CTRL_28	0000_0000	STCU	None
000_0000_0000_0100	1_1001_1101	STCU_MB_CTRL_29	0000_0000	STCU	None
000_0000_0000_0100	1_1001_1110	STCU_MB_CTRL_30	0000_0000	STCU	None
000_0000_0000_0100	1_1001_1111	STCU_MB_CTRL_31	0000_0000	STCU	None
000_0000_0000_0100	1_1010_0000	STCU_MB_CTRL_32	0000_0000	STCU	None
000_0000_0000_0100	1_1010_0001	STCU_MB_CTRL_33	0000_0000	STCU	None
000_0000_0000_0100	1_1010_0010	STCU_MB_CTRL_34	0000_0000	STCU	None
000_0000_0000_0100	1_1010_0011	STCU_MB_CTRL_35	0000_0000	STCU	None
000_0000_0000_0100	1_1010_0100	STCU_MB_CTRL_36	0000_0000	STCU	None
000_0000_0000_0100	1_1010_0101	STCU_MB_CTRL_37	0000_0000	STCU	None
000_0000_0000_0100	1_1010_0110	STCU_MB_CTRL_38	0000_0000	STCU	None
000_0000_0000_0100	1_1010_0111	STCU_MB_CTRL_39	0000_0000	STCU	None
000_0000_0000_0100	1_1010_1000	STCU_MB_CTRL_40	0000_0000	STCU	None
000_0000_0000_0100	1_1010_1001	STCU_MB_CTRL_41	0000_0000	STCU	None
000_0000_0000_0100	1_1010_1010	STCU_MB_CTRL_42	0000_0000	STCU	None
000_0000_0000_0100	1_1010_1011	STCU_MB_CTRL_43	0000_0000	STCU	None
000_0000_0000_0100	1_1010_1100	STCU_MB_CTRL_44	0000_0000	STCU	None
000_0000_0000_0100	1_1010_1101	STCU_MB_CTRL_45	0000_0000	STCU	None
000_0000_0000_0100	1_1010_1110	STCU_MB_CTRL_46	0000_0000	STCU	None
000_0000_0000_0100	1_1010_1111	STCU_MB_CTRL_47	0000_0000	STCU	None
000_0000_0000_0100	1_1011_0000	STCU_MB_CTRL_48	0000_0000	STCU	None
000_0000_0000_0100	1_1011_0001	STCU_MB_CTRL_49	0000_0000	STCU	None
000_0000_0000_0100	1_1011_0010	STCU_MB_CTRL_50	0000_0000	STCU	None

Table continues on the next page...

Table 5-6. UTEST DCF clients (continued)

DCF CS[14:0]	DCF Address[10:2]	DCF client description	DCF client reset value (hex)	DCF client target module	DCF client special strategy
000_0000_0000_0100	1_1011_0011	STCU_MB_CTRL_51	0000_0000	STCU	None
000_0000_0000_0100	1_1011_0100	STCU_MB_CTRL_52	0000_0000	STCU	None
000_0000_0000_0100	1_1011_0101	STCU_MB_CTRL_53	0000_0000	STCU	None
000_0000_0000_0100	1_1011_0110	STCU_MB_CTRL_54	0000_0000	STCU	None
000_0000_0000_0100	1_1011_0111	STCU_MB_CTRL_55	0000_0000	STCU	None
000_0000_0000_0100	1_1011_1000	STCU_MB_CTRL_56	0000_0000	STCU	None
000_0000_0000_0100	1_1011_1001	STCU_MB_CTRL_57	0000_0000	STCU	None
000_0000_0000_0100	1_1011_1010	STCU_MB_CTRL_58	0000_0000	STCU	None
000_0000_0000_0100	1_1011_1011	STCU_MB_CTRL_59	0000_0000	STCU	None
000_0000_0000_0100	1_1011_1100	STCU_MB_CTRL_60	0000_0000	STCU	None
000_0000_0000_0100	1_1011_1101	STCU_MB_CTRL_61	0000_0000	STCU	None
000_0000_0000_0100	1_1011_1110	STCU_MB_CTRL_62	0000_0000	STCU	None
000_0000_0000_0100	1_1011_1111	STCU_MB_CTRL_63	0000_0000	STCU	None
000_0000_0000_0100	1_1100_0000	STCU_MB_CTRL_64	0000_0000	STCU	None
PASS					
000_0000_0000_1000	0_0000_0001-0_0010_1011	Reserved			
000_0000_0000_1000	0_0010_1100	Censorship	0000_0000	PASS	None
000_0000_0000_1000	0_0010_1101-0_0010_1111	Reserved			
000_0000_0000_1000	0_0011_0000	Production Disable	0000_0000	PASS	write_once
000_0000_0000_1000	0_0100_0000	LOCK0_PG0	FFFF_FFFF	PASS	None
000_0000_0000_1000	0_0100_0001	LOCK1_PG0	FFFF_FFFF	PASS	None
000_0000_0000_1000	0_0100_0010	LOCK2_PG0	FFFF_FFFF	PASS	None
000_0000_0000_1000	0_0100_0011	LOCK3_PG0	FFFF_FFFF	PASS	None
000_0000_0000_1000	0_0100_0100	LOCK0_PG1	FFFF_FFFF	PASS	None
000_0000_0000_1000	0_0100_0101	LOCK1_PG1	FFFF_FFFF	PASS	None
000_0000_0000_1000	0_0100_0110	LOCK2_PG1	FFFF_FFFF	PASS	None
000_0000_0000_1000	0_0100_0111	LOCK3_PG1	FFFF_FFFF	PASS	None
000_0000_0000_1000	0_0100_1000	LOCK0_PG2	FFFF_FFFF	PASS	None
000_0000_0000_1000	0_0100_1001	LOCK1_PG2	FFFF_FFFF	PASS	None
000_0000_0000_1000	0_0100_1010	LOCK2_PG2	FFFF_FFFF	PASS	None
000_0000_0000_1000	0_0100_1011	LOCK3_PG2	FFFF_FFFF	PASS	None
000_0000_0000_1000	0_0100_1100	LOCK0_PG3	FFFF_FFFF	PASS	None
000_0000_0000_1000	0_0100_1101	LOCK1_PG3	FFFF_FFFF	PASS	None
000_0000_0000_1000	0_0100_1110	LOCK2_PG3	FFFF_FFFF	PASS	None
000_0000_0000_1000	0_0100_1111	LOCK3_PG3	FFFF_FFFF	PASS	None
000_0000_0000_1000	0_0101_0000-0_1111_1111	Reserved			
Tamper Detect (TDM)					

Table continues on the next page...

Table 5-6. UTEST DCF clients (continued)

DCF CS[14:0]	DCF Address[10:2]	DCF client description	DCF client reset value (hex)	DCF client target module	DCF client special strategy
000_0000_0001_0000	0_0000_0000	Diary Base Address	FFFF_FFFC	TDM	write_once + wr0_only
000_0000_0001_0000	0_0000_0001	Tamper Region Override	0000_0000	TDM	wr1_only
000_0000_0001_0000	0_0000_0010	Software Region Override Disable	0000_0000	TDM	write_once + wr1_only
000_0000_0001_0000	0_0000_0011-0_0000_0111	Reserved			
000_0000_0001_0000	0_0000_1000	OTP_EN0	0000_0000	TDM	wr1_only
000_0000_0001_0000	0_0000_1001	OTP_EN1	0000_0000	TDM	wr1_only
000_0000_0001_0000	0_0000_1010	OTP_EN2	0000_0000	TDM	wr1_only
000_0000_0001_0000	0_0000_1011	OTP_EN3	0000_0000	TDM	wr1_only
000_0000_0001_0000	0_0000_1100-0_0001_0011	Reserved			
000_0000_0001_0000	0_0001_0100	TDR0_LOCK0	0000_0000	TDM	wr1_only
000_0000_0001_0000	0_0001_0101	TDR0_LOCK1	0000_0000	TDM	wr1_only
000_0000_0001_0000	0_0001_0110	TDR0_LOCK2	0000_0000	TDM	wr1_only
000_0000_0001_0000	0_0001_0111	TDR0_LOCK3	0000_0000	TDM	wr1_only
000_0000_0001_0000	0_0001_1000	TDR1_LOCK0	0000_0000	TDM	wr1_only
000_0000_0001_0000	0_0001_1001	TDR1_LOCK1	0000_0000	TDM	wr1_only
000_0000_0001_0000	0_0001_1010	TDR1_LOCK2	0000_0000	TDM	wr1_only
000_0000_0001_0000	0_0001_1011	TDR1_LOCK3	0000_0000	TDM	wr1_only
000_0000_0001_0000	0_0001_1100	TDR2_LOCK0	0000_0000	TDM	wr1_only
000_0000_0001_0000	0_0001_1101	TDR2_LOCK1	0000_0000	TDM	wr1_only
000_0000_0001_0000	0_0001_1110	TDR2_LOCK2	0000_0000	TDM	wr1_only
000_0000_0001_0000	0_0001_1111	TDR2_LOCK3	0000_0000	TDM	wr1_only
000_0000_0001_0000	0_0010_0000	TDR3_LOCK0	0000_0000	TDM	wr1_only
000_0000_0001_0000	0_0010_0001	TDR3_LOCK1	0000_0000	TDM	wr1_only
000_0000_0001_0000	0_0010_0010	TDR3_LOCK2	0000_0000	TDM	wr1_only
000_0000_0001_0000	0_0010_0011	TDR3_LOCK3	0000_0000	TDM	wr1_only
000_0000_0001_0000	0_0010_0100	TDR4_LOCK0	0000_0000	TDM	wr1_only
000_0000_0001_0000	0_0010_0101	TDR4_LOCK1	0000_0000	TDM	wr1_only
000_0000_0001_0000	0_0010_0110	TDR4_LOCK2	0000_0000	TDM	wr1_only
000_0000_0001_0000	0_0010_0111	TDR4_LOCK3	0000_0000	TDM	wr1_only
000_0000_0001_0000	0_0010_1000	TDR5_LOCK0	0000_0000	TDM	wr1_only
000_0000_0001_0000	0_0010_1001	TDR5_LOCK1	0000_0000	TDM	wr1_only
000_0000_0001_0000	0_0010_1010	TDR5_LOCK2	0000_0000	TDM	wr1_only
000_0000_0001_0000	0_0010_1011	TDR5_LOCK3	0000_0000	TDM	wr1_only
000_0000_0001_0000	0_0010_1100-0_1111_1111	Reserved			
MISCELLANEOUS					

Table continues on the next page...

Table 5-6. UTEST DCF clients (continued)

DCF CS[14:0]	DCF Address[10:2]	DCF client description	DCF client reset value (hex)	DCF client target module	DCF client special strategy
000_0000_1000_0000	0_0000_0000	UTEST Miscellaneous	B008_01C0	Several ¹	Triple Voted ²
000_0000_1000_0000	0_0000_0001	UTEST Secure Boot Configuration	0000_000D	CSE	write_once (by NXP)
000_0000_1000_0000	0_0000_0010	UTEST Secure Boot Code Address	0000_0000	CSE	None
000_0000_1000_0000	0_0000_0011	UTEST Secure Boot Code Length	0000_0000	CSE	None
000_0000_1000_0000	0_0000_0100	UTEST XOSC ANA	0000_0004	XOSC Analog	None ²
000_0000_1000_0000	0_0000_0101	UTEST ERROR Functions Config	0000_0000		Triple Voted
000_0000_1000_0000	0_0000_0110	UTEST Reset Escalation	0000_0000	SIU	Triple Voted
000_0000_1000_0000	0_0000_0111-0_0001_1110	Reserved			
000_0000_1000_0000	0_0001_1111	Reset Event Enable Control	0000_00FF	PMC Digital	Triple Voted + Parity
000_0000_1000_0000	0_0010_0000	Temperature Sensor Reset Event Enable Control	0000_003F	PMC Digital	Triple Voted + Parity
000_0000_1000_0000	0_0010_0001	Reserved			
000_0000_1000_0000	0_0010_0010	VREG Enable	0000_0000	PMC Digital	Triple Voted + Parity ²
000_0000_1000_0000	0_0010_0011-0_1111_1111	Reserved			

1. See Miscellaneous DCF client description

2. Be careful when programming the UTEST Miscellaneous, XOSC ANA, and VREG Enable DCF clients. DCF records are processed sequentially in the order they are read from flash memory. DCF records that control actions such as enabling/disabling voltage regulators or detectors should not be programmed more than once with different values.

5.7.1 UTEST Miscellaneous DCF client

This is a 32-bit Triple-Voting DCF Client that implements a variety of settings related to different modules and functionalities, such as boot settings on SIU, oscillator trimming, oscillator enable/disable, and lockstep mode selection. The reset value indicated below is the value when no DCF record has been read or written.

NOTE

Be careful when programming the UTEST Miscellaneous, XOSC ANA, and VREG Enable DCF clients. DCF records are processed sequentially in the order they are read from flash

memory. DCF records that control actions such as enabling/disabling voltage regulators or detectors should not be programmed more than once with different values.

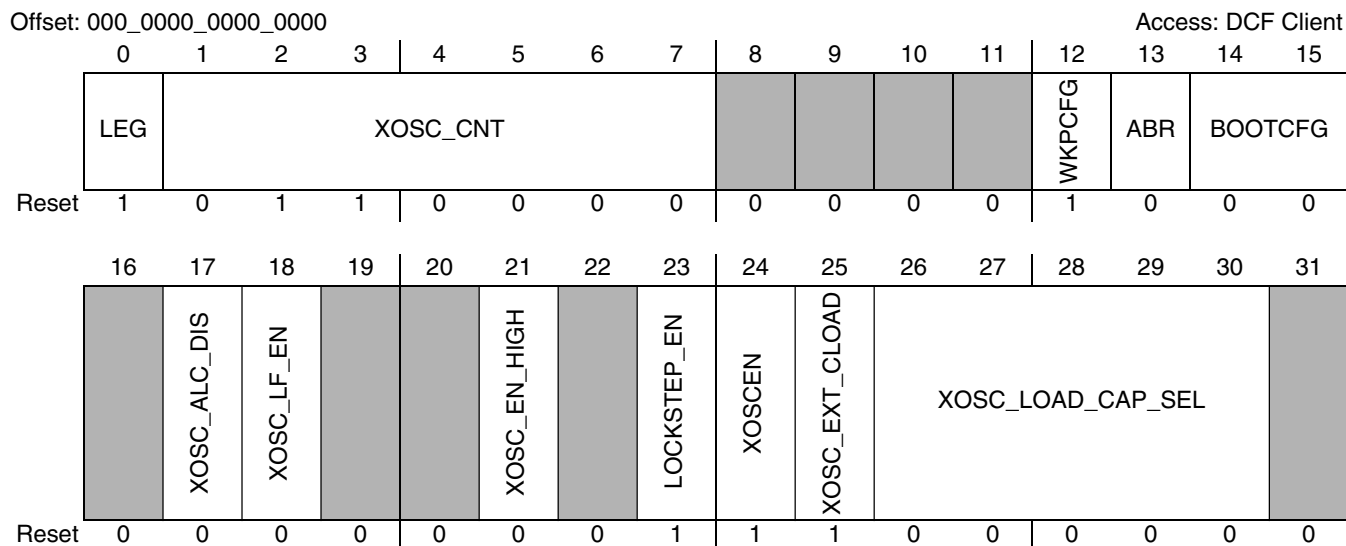


Figure 5-4. UTEST Miscellaneous DCF client

Table 5-7. UTEST Miscellaneous DCF client field descriptions

Field	Description
0 LEG	Legacy Mode. This bit indicates whether the WKPCFG, ABR, and BOOTCFG settings should come from package pins or from the Miscellaneous DCF client. 0 Settings come from this DCF client 1 Settings driven from pins
1-7 XOSC_CNT	Oscillator Stabilization Counter. These seven bits are used to pre-load a counter clocked by the crystal oscillator. Upon reset negation, the counter runs for 512 x XOSC_CNT clock cycles, then sets the XOSC flag in the SIU_RSR register.
8-11 Reserved	Reserved
12 WKPCFG	Weak Pull Configuration. SIU_RSR stores this configuration during reset. This information is then used to configure pull up/down of external pins. 0 Weak pulldown is selected 1 Weak pullup is selected
13 ABR	Auto Baud Rate. SIU_RSR stores this configuration during reset. This information is then used by the Boot Code to configure the Auto Baud Rate feature. 0 Auto Baud Rate disabled 1 Auto Baud Rate enabled

Table continues on the next page...

Table 5-7. UTEST Miscellaneous DCF client field descriptions (continued)

Field	Description
14-15 BOOTCFG	Boot Mode Selection. SIU_RSR stores this configuration during reset. This information is then used to by the Boot Code to configure the boot sequence. 00 Internal boot mode 01 Serial boot mode 10 Non-Muxed EBI boot mode (16-bit data, non-muxed address bus) 11 Muxed EBI boot mode (16-bit data, muxed address bus)
16 Reserved	Reserved
17 XOSC_ALC_DIS	Oscillator Automatic Level Controller 0 Automatic level controller enabled 1 Automatic level controller disabled
18 XOSC_LF_EN	Oscillator Low Setting Selection Set this bit to 1 only when XOSC_EN_HIGH is 0. Setting this bit to 1 when XOSC_EN_HIGH is 1 selects a high setting, but this combination is not recommended.¹ 0 Oscillator is used for medium or high setting 1 Oscillator is used for low setting
19-20 Reserved	Reserved
21 XOSC_EN_HIGH	Oscillator Medium or High Setting Selection If the LEG bit is 0, this field is loaded by the SSCM during reset into the XOSCHIGH bit of SIU_RSR. This bit determines whether the oscillator operates at medium or high setting. Use this bit to select between medium or high setting when XOSC_LF_EN is 0. Setting this bit to 1 when XOSC_LF_EN is 1 selects a high setting, but this combination is not recommended.¹ 0 Oscillator at medium setting 1 Oscillator at high setting
22 Reserved	Reserved
23 LOCKSTEP_EN	Lockstep Mode. This bit is used to enable the Safety Core to be in lockstep with the Core1. 0 Lockstep disabled 1 Lockstep enabled
24 XOSCEN	Crystal Oscillator Enable. This bit is used to enable the crystal oscillator. If disabled, an external clock source can be connected to the EXTAL pin, with the oscillator working in bypass mode. 0 Crystal oscillator disabled, bypass mode with clock sourced from EXTAL pin. 1 Oscillator enabled, crystal connected.
25 XOSC_EXT_CLOAD	Oscillator External Cap 0 Selects oscillator internal cap: acceptable for oscillators that require load capacitance of 10 pF or less 1 Selects oscillator external cap: required for oscillators that have load capacitance of 10 pF or greater

Table continues on the next page...

Table 5-7. UTEST Miscellaneous DCF client field descriptions (continued)

Field	Description
26-30 XOSC_LOAD_CAP_SEL	Oscillator Load Capacitance Selection. Five bits to program the load capacitance. Load capacitor values are determined by the crystal manufacturer data sheet.

1. "Low" is the setting commonly used for crystals at 8 MHz, "medium" is commonly used for crystals greater than 8 MHz to 20 MHz, and "high" is commonly used for crystals greater than 20 MHz to 40 MHz. However, the user must characterize carefully to determine the best g_m setting for the intended application because crystal load capacitance, board layout, and other factors affect the g_m value that is needed. See the Data Sheet for additional information.

5.7.2 UTEST Secure Boot Configurations DCF client

This DCF client, in conjunction with the UTEST Secure Boot Code Address and UTEST Secure Boot Code Length clients, configures the CSE security module to carry out a secure boot in the device. This 32-bit write-once DCF client selects:

- whether the CSE module is enabled
- whether the secure boot is to be executed
- the mode (sequential or parallel) in which the secure boot runs
- whether CSE must be disabled in the Failure Analysis (FA) life cycle

The CSE module must be enabled for the secure boot to execute.

When this client selects to execute the secure boot, the boot occurs during reset on the memory space defined by the Secure Boot Code Address and Code Length DCF clients.

NOTE

This write-once DCF client is programmed at the factory with the value 0000_000Dh. This value cannot be overwritten. This configuration:

- enables CSE
- enables CSE to execute the secure boot during reset
- configures the secure boot to occur in parallel mode
- disables CSE when the life cycle advances to FA

Offset: 000_0000_0000_0001												Access: DCF Client / Write Once				
	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
Reset	0	0	0	0	0	0	0	0	0	0	0	0	FAD	SBE	SBM	MEN
	0	0	0	0	0	0	0	0	0	0	0	0	1	1	0	1

Figure 5-5. UTEST Secure Boot Configurations DCF client

Table 5-8. UTEST Secure Boot Configurations DCF client field descriptions

Field	Description
0-27 Reserved	Reserved
28 FAD	CSE FA Disable Set this field to 1 to disable the CSE module when the chip's life cycle matures to Failure Analysis (FA). When CSE boots, security information such as the Security Keys is always loaded into RAM. This bit is meant to provide some protection against an MBIST-based attack to extract that security information from the RAM. 0 CSE module will not be disabled when device is matured to FA. 1 CSE module will be disabled when device is matured to FA. NOTE: If the MEN bit is set to 1, the CSE module will be disabled in FA when FAD is 1, but in FA the CSE will still maintain exclusive access to the flash memory blocks mapped to the C55FMC_LOCK1 register, PASS_LOCK1_PGn registers, and TDRn_LOCK1 DCF client.
29 SBE	Secure Boot Enable 0 Secure boot disabled. CSE will not execute the secure boot during reset. 1 Secure boot enabled. CSE will execute the secure boot during reset in parallel or serial mode as per SBM settings.
30 SBM	Secure Boot Mode 0 Secure boot runs in parallel mode. In this mode, CSE executes the secure boot in parallel with the user boot code. 1 Secure boot runs in sequential mode. In this mode, the user boot mode starts only after CSE finishes the secure boot.
31 MEN	CSE Module Enable This bit indicates that the CSE module will be used in the part. 0 CSE is not used. The CSE clock is disabled, and accesses to the registers generate a bus error. 1 CSE is enabled. NOTE: The secure boot functionality works only if MEN is set to 1. The secure boot functionality does not work if CSE is disabled.

5.7.3 UTEST Secure Boot Code Address DCF client

This DCF client, in conjunction with the UTEST Secure Boot Configurations and UTEST Secure Boot Code Length clients, configures the CSE security module to carry out a secure boot in the device. This 32-bit DCF client provides CSE with the location in memory of the boot code to be verified.

DCF clients

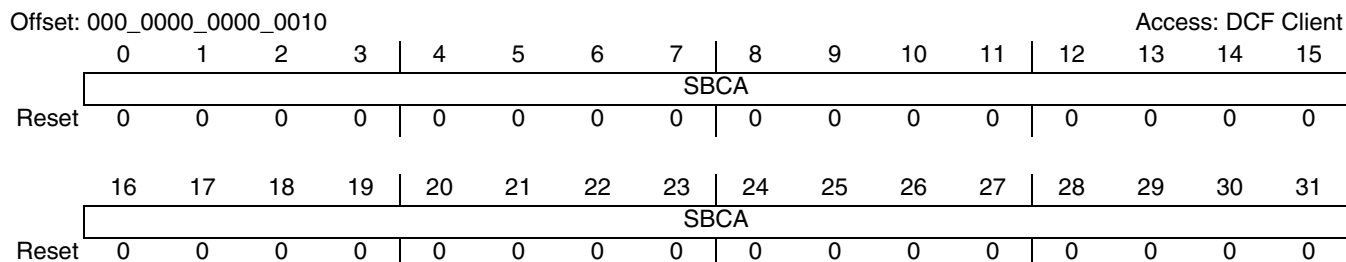


Figure 5-6. UTEST Secure Boot Code Address DCF client

Table 5-9. UTEST Secure Boot Code Address DCF client field descriptions

Field	Description
0-31	Secure Boot Code Address
SBCA	This 32-bit field indicates the start address of the secure boot code.

5.7.4 UTEST Secure Boot Code Length DCF client

This DCF client, in conjunction with the UTEST Secure Boot Configurations and UTEST Secure Boot Code Address clients, configures the CSE security module to carry out a secure boot in the device. This 32-bit DCF client provides CSE with the length of the secure boot code.

NOTE

Users whose applications do not use the CSE module, which is automatically enabled, should ensure that the Secure Boot Code Length DCF client has the value 0000_0000h (its default value).

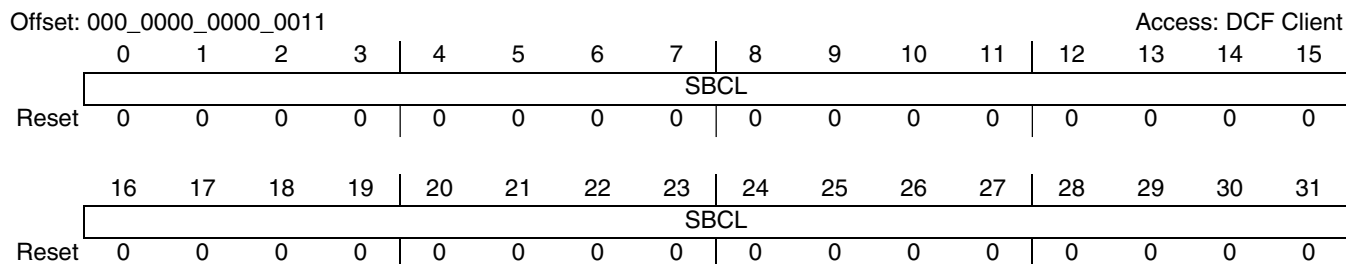


Figure 5-7. UTEST Secure Boot Code Length DCF client

Table 5-10. UTEST Secure Boot Code Length DCF client field descriptions

Field	Description
0-31	Secure Boot Code Length
SBCL	This 32-bit field indicates the size in bytes of the secure boot code.

5.7.5 UTEST Analog External Oscillator (XOSC ANA) DCF client

This 32-bit DCF client is used to apply settings to the analog external oscillator. The internal current control setting is the only feature available and is used to boot the current. This setting is accessible in functional mode for test and debug purposes.

NOTE

Be careful when programming the UTEST Miscellaneous, XOSC ANA, and VREG Enable DCF clients. DCF records are processed sequentially in the order they are read from flash memory. DCF records that control actions such as enabling/disabling voltage regulators or detectors should not be programmed more than once with different values.

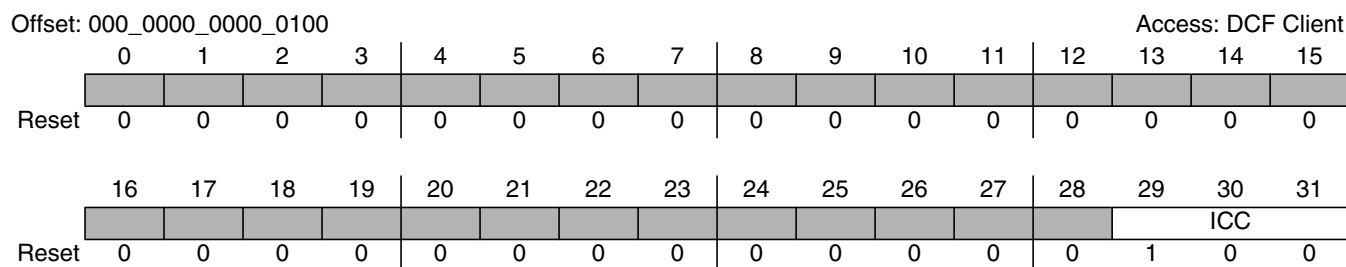


Figure 5-8. UTEST Analog External Oscillator DCF client

Table 5-11. UTEST Analog External Oscillator DCF client field descriptions

Field	Description
0-28 Reserved	Reserved
29-31 ICC	Internal Current Control These bits allow the internal current to be disabled (no current flowing, so the oscillator does not work), decreased, and increased. 000 No (0x) current 001 0.25x: decrease current by 25% of default current 010 0.5x: decrease current by 50% of default current 100 1x: default current, which is 10 mA 101 1.25x: increase current by 25% of default current 110 1.5x: increase current by 50% of default current 111 1.75x: increase current by 75% of default current

5.7.6 UTEST Error Functions Pads Configuration DCF client

This 32-bit Triple-Voting DCF client enables the ERROR function on the ERROR0, ERROR1, and ERRORIN pins independently.

CAUTION

The ERROR function is disabled by the DCF at power-on reset (POR). After it is enabled, the ERROR function is locked on the pin until the next POR. Any other type of reset has no effect on these settings.

Offset: 000_0000_0000_0101																Access: DCF Client					
	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15					
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0					
	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31					
Reset	0	0	0	0	0	0	0	0	0	0	0	0	E1OD	E0OD	EINE	E1E	E0E				

Figure 5-9. UTEST Error Functions Pads Configuration DCF client

Table 5-12. UTEST Error Functions Pads Configuration DCF client field descriptions

Field	Description
0-26 Reserved	Reserved
27 E1OD	ERROR1 Open Drain Enable Selects open-drain or push-pull output configuration on ETPUA11 pin. Valid only if E1E is set to 1. 0 Push-pull output selection 1 Open-drain output selection
28 E0OD	ERROR0 Open Drain Enable Selects open-drain or push-pull output configuration on ETPUA3 pin. Valid only if E0E is set to 1. 0 Push-pull output selection 1 Open-drain output selection
29 EINE	ERRORIN Enable Selects multiplexing of ETPUA10 pin. 0 ETPUA10 pin function depends on PCR124 settings 1 Enables and locks ERRORIN function on ETPUA10 pin
30 E1E	ERROR1 Enable Selects multiplexing of ETPUA11 pin. 0 ETPUA11 pin function depends on PCR125 settings 1 Enables and locks ERROR1 function on ETPUA11 pin

Table continues on the next page...

Table 5-12. UTEST Error Functions Pads Configuration DCF client field descriptions (continued)

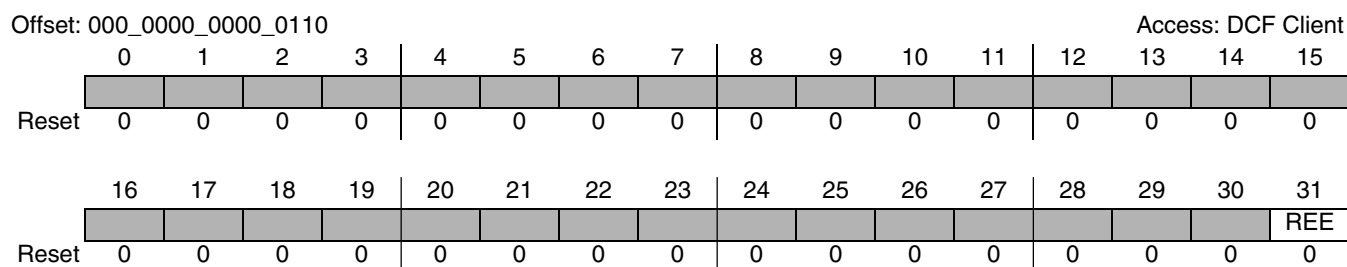
Field	Description
31 E0E	ERROR0 Enable Selects multiplexing of ETPUA3 pin. 0 ETPUA3 pin function depends on PCR117 settings 1 Enables and locks ERROR0 function on ETPUA3 pin

5.7.7 UTEST Reset Escalation Configuration DCF client

This 32-bit Triple-Voting DCF client enables the Reset Escalation function on the reset sequence of the device. See the Reset chapter for further details on this feature.

CAUTION

The Reset Escalation function is disabled by the DCF at power-on reset (POR). After it is enabled, the Reset Escalation function is locked until the next POR. Any other type of reset has no effect on these settings.

**Figure 5-10. UTEST Reset Escalation Configuration DCF client****Table 5-13. UTEST Reset Escalation Configuration DCF client field descriptions**

Field	Description
0-30 Reserved	Reserved
31 REE	Reset Escalation Enable Enables a reset escalation counter that is incremented at the end of every reset cycle. If the counter exceeds the number programmed in SIU_RCR[RET], the device is locked into a reset state that can be undone only by a power-up cycle. NOTE: LVD/HVD and temperature events that are enabled in the PMC to trigger a reset cause a POR. In these cases, the reset escalation counter resets. 0 Reset Escalation disabled 1 Reset Escalation enabled

5.7.8 UTEST PMC Reset Enable Control (REE CTRL) DCF client

This DCF client trims the PMC_REE register. See the PMC chapter for details about this register.

Offset: 0000_0000_0001_1111

5.7.9 UTEST PMC Temperature Sensor Reset Event Enable Control (REE TMPSNS) DCF client

This DCF client trims the PMC_REE_TD register. See the PMC chapter for details about this register.

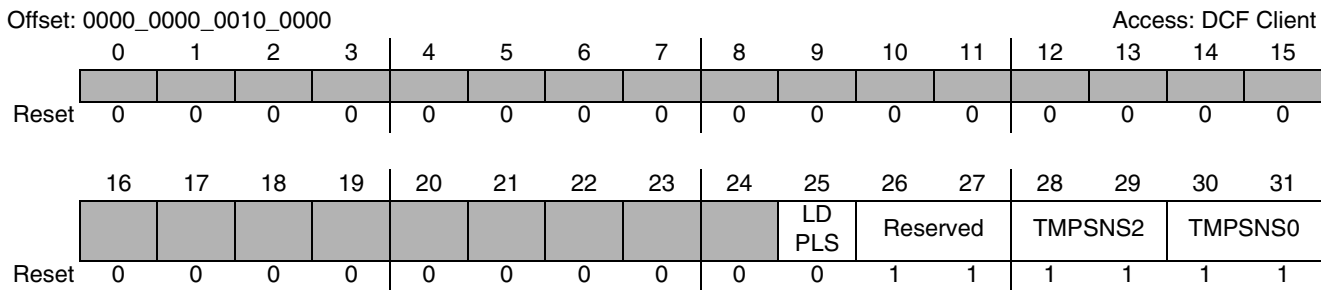


Figure 5-11. UTEST PMC Temperature Sensor Reset Event Enable Control DCF client

Table 5-14. UTEST PMC Temperature Sensor Reset Event Enable Control DCF client field descriptions

Field	Description
0-24 Reserved	Reserved
25 LDPLS	Load pulse to latch Temperature Sensor REE values 0 Load the PMC_REE_TD register with IPS bus data. 1 Load the PMC_REE_TD register with the data in this DCF client.
26-27 Reserved	Reserved
28-29 TMPSNS2	For both Temperature Sensor modules 0 and 1, REE control of temperature 2: 150°C
30-31 TMPSNS0	For both Temperature Sensor modules 0 and 1, REE control of temperature 0: -40°C

5.7.10 UTEST PMC Voltage Regulator Enable (VREG ENB) DCF client

This 32-bit DCF client implements the active low enables for three PMC voltage regulators.

NOTE

Be careful when programming the UTEST Miscellaneous, XOSC ANA, and VREG Enable DCF clients. DCF records are processed sequentially in the order they are read from flash memory. DCF records that control actions such as enabling/disabling voltage regulators or detectors should not be programmed more than once with different values.

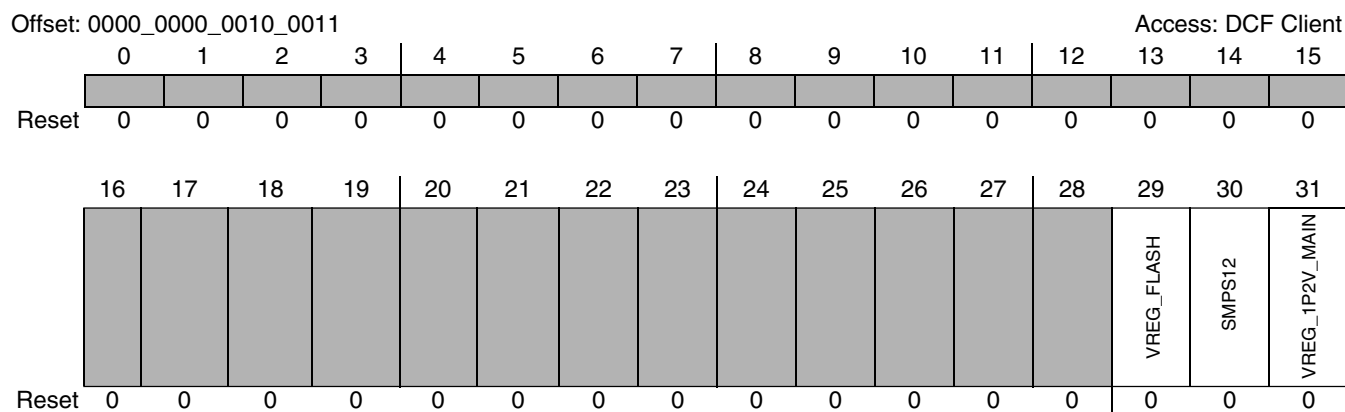


Figure 5-12. UTEST PMC Voltage Regulator Enable DCF client

Table 5-15. UTEST PMC Voltage Regulator Enable DCF client field descriptions

Field	Description
0-28 Reserved	Reserved
29 VREG_FLASH	Flash memory regulator enable 0: Enabled 1: Disabled
30 SMPS12	Switched Mode Power Supply (SMPS) enable 0: Enabled 1: Disabled
31 VREG_1P2V_MAIN	Core regulator enable 0: Enabled 1: Disabled

Chapter 6

e200z7 Core Complex Overview

6.1 Core complex

The following table summarizes the cores on this chip.

Table 6-1. Core complex overview

Core identifier	Core type	Brief description
Core0	e200z759n3	Main boot computational core
Core1	e200z759n3	Duplicated computational core
Checker core	e200z758	Operates in delayed lockstep with Core1, executing exactly the same instructions to confirm correct execution

6.2 Core0 and Core1 overview

The two e200z759n3 cores are identical.

For a comprehensive core description, see the *e200z759n3 Core Reference Manual* (document number e200z759n3CRM).

See [End-to-end Error Correction Code \(e2eECC\)](#) for any limitations of the cores due to the implementation of e2eECC.

6.3 Checker core overview

The e200z758 core does not have these features of the e200z759n3 computational cores:

- Instruction cache and data cache
- Nexus 3+ module

Otherwise, the e200z758 checker core has the same features as the e200z759n3 cores.

Chapter 7

Platform Coherency Unit (PCU)

7.1 Introduction

The Platform Coherency Unit is a hardware mechanism for maintaining memory coherency between multiple bus masters. It supports cache coherency between the two cores, including writes via the Nexus read-write access block. In addition, DMA writes to system memory may be configured to provide cache coherency.

For writes originating from either core, the MMU for the core must be configured with Memory Coherence (M bit) and Write Through (W bit) set for the address spaces to be accessed. For writes originating from either DMA module, use the [Global PCU Output Register \(DMA_GPOR_n\)](#) to enable coherency on a channel-by-channel basis. Finally, set the Master Write Monitor Enable bits for each master, requiring coherency for its writes, and the global enable bit (ENB) in the PCU's configuration register.

NOTE

On predecessor devices, the Platform Coherency Unit (PCU) was named the Cache Coherency Unit (CCU).

7.2 Memory map and register descriptions

The programming model contains control and status registers. It can be referenced using only 32-bit (word) accesses. Writes to the configuration fields can be performed only in supervisor mode. Attempted accesses using non-32-bit data sizes, accesses to undefined or reserved addresses, or accesses with a non-supported access type result in a bus error termination.

PCU memory map

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
0	PCU Configuration and Error Status Register (PCU_CESR)	32	R/W	0300_0000h	7.2.1/172
10	PCU Error Address Register, Core n (PCU_EAR0)	32	R	0000_0000h	7.2.2/176
14	PCU Error Detail Register, Core n (PCU_EDR0)	32	R	0000_0000h	7.2.3/177
20	PCU Error Address Register, Core n (PCU_EAR1)	32	R	0000_0000h	7.2.2/176
24	PCU Error Detail Register, Core n (PCU_EDR1)	32	R	0000_0000h	7.2.3/177
30	PCU Interrupt Register, Core n (PCU_IR0)	32	R/W	0000_0000h	7.2.4/178
34	PCU Interrupt Register, Core n (PCU_IR1)	32	R/W	0000_0000h	7.2.4/178

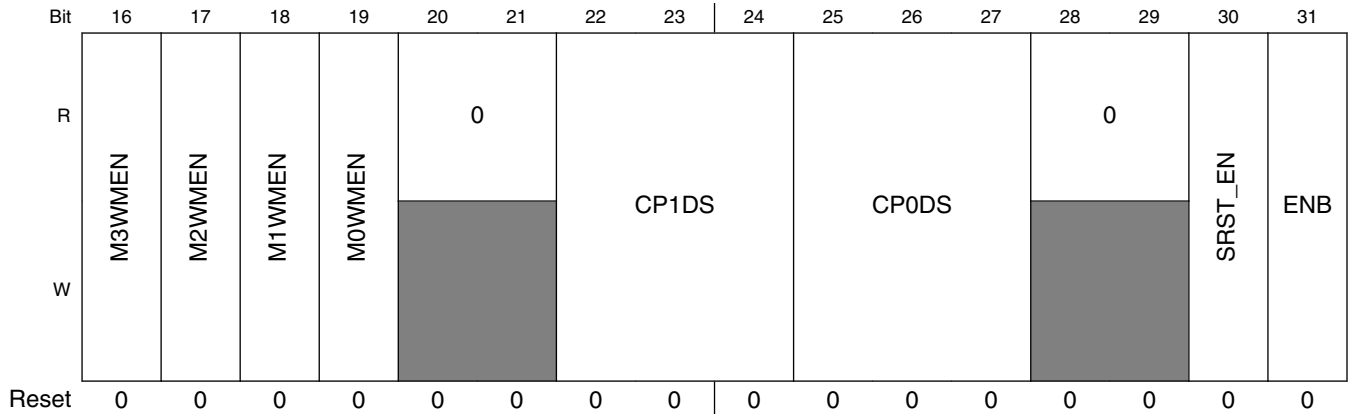
7.2.1 PCU Configuration and Error Status Register (PCU_CESR)

Access by mode:

- Read: all modes
- Write: Supervisor mode

Address: 0h base + 0h offset = 0h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	CP1ERR[1:0]		CP0ERR[1:0]		SRST_FLAG	0	CP1IDLE	CP0IDLE	0		CP1IEN	CP0IEN	0			
W	w1c		w1c													
Reset	0	0	0	0	0	0	1	1	0	0	0	0	0	0	0	0



PCU_CESR field descriptions

Field	Description
0–1 CP1ERR[1:0]	Core Processor 1 Error This field represents a flag maintained by the PCU for signaling the presence of a captured snoop error or the occurrence of a queue overflow. The error details are contained in PCU_EAR1 and PCU_EDR1. CP1ERR[1] is set when the snoop queue for Core1 has overflowed; the address and attributes of the global write that caused the queue overflow are recorded in PCU_EAR1 and PCU_EDR1. CP1ERR[0] is set when the hardware detects a snoop error and records the faulting address and attributes. This field is cleared when the corresponding bits are written as 1. If another Core1 error occurs before the CP1ERR bit is cleared, CP1ERR remains set while PCU_EAR1 and PCU_EDR1 capture address and attributes for the most recent error event. 00 PCU_EAR1/PCU_EDR1 do not contain a captured snoop error. 01 PCU_EAR1/PCU_EDR1 contain a captured snoop error. 10 PCU_EAR1/PCU_EDR1 contain a queue overflow occurrence. 11 A snoop error event occurred, followed by a queue overflow event. Because the PCU_EAR1/PCU_EDR1 registers capture the most recent error event, PCU_EAR1/PCU_EDR1 contain the queue overflow occurrence.
2–3 CP0ERR[1:0]	Core Processor 0 Error This field represents a flag maintained by the PCU for signaling the presence of a captured snoop error or the occurrence of a queue overflow. The error details are contained in PCU_EAR0 and PCU_EDR0. CP0ERR[1] is set when the snoop queue for Core0 has overflowed; the address and attributes of the global write that caused the queue overflow are recorded in PCU_EAR0 and PCU_EDR0. CP0ERR[0] is set when the hardware detects a snoop error and records the faulting address and attributes. This field is cleared when the corresponding bits are written as 1. If another Core0 error occurs before the CP0ERR bit is cleared, CP0ERR remains set while PCU_EAR0 and PCU_EDR0 capture address and attributes for the most recent error event. 00 PCU_EAR0/PCU_EDR0 do not contain a captured snoop error. 01 PCU_EAR0/PCU_EDR0 contain a captured snoop error. 10 PCU_EAR0/PCU_EDR0 contain a queue overflow occurrence. 11 A snoop error event occurred, followed by a queue overflow event. Because the PCU_EAR0/PCU_EDR0 registers capture the most recent error event, PCU_EAR0/PCU_EDR0 contain the queue overflow occurrence.
4 SRST_FLAG	Software Reset Flag This field represents a flag maintained by the PCU to indicate that a software reset event has occurred. A software reset of the PCU is invoked by programming the PCU_CESR[SRST_EN] field.

Table continues on the next page...

PCU_CESR field descriptions (continued)

Field	Description
	0 The last PCU reset event was a hardware reset event. 1 The last PCU reset event was a software-invoked reset event.
5 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
6 CP1IDLE	Core Processor 1 Idle This read-only status flag indicates when the Core1 snoop queue is empty. 0 Core1 snoop queue is not empty; Core1 snoop queue is busy processing valid snoop requests to Core1. 1 Core1 snoop queue is empty.
7 CP0IDLE	Core Processor 0 Idle This read-only status flag indicates when the Core0 snoop queue is empty. 0 Core0 snoop queue is not empty; Core0 snoop queue is busy processing valid snoop requests to Core0. 1 Core0 snoop queue is empty.
8–9 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
10 CP1IEN	Core Processor 1 Interrupt Enable This bit enables interrupt generation upon detection of a Core1 snoop request that results in an error. 0 PCU generates an interrupt request upon detection of a Core1 snoop that terminated with error. 1 PCU does not generate an interrupt request upon detection of a Core1 snoop request that terminated with error.
11 CP0IEN	Core Processor 0 Interrupt Enable This bit enables interrupt generation upon detection of a Core0 snoop request that results in an error. 0 PCU generates an interrupt request upon detection of a Core0 snoop that terminated with error. 1 PCU does not generate an interrupt request upon detection of a Core0 snoop request that terminated with error.
12–15 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
16 M3WMEN	Master 3 Write Monitor Enable This bit enables monitoring of eDMA_B for global writes. 0 Disable eDMA_B monitoring. Global writes from eDMA_B do not initiate any snoop requests. 1 Enable eDMA_B monitoring for global writes.
17 M2WMEN	Master 2 Write Monitor Enable This bit enables monitoring of eDMA_A for global writes. 0 Disable eDMA_A monitoring. Global writes from eDMA_A do not initiate any snoop requests. 1 Enable eDMA_A monitoring for global writes.
18 M1WMEN	Master 1 Write Monitor Enable This bit enables monitoring of Core1 for global writes.

Table continues on the next page...

PCU_CESR field descriptions (continued)

Field	Description
	0 Disable Core1 monitoring. Global writes from Core1 do not initiate any snoop requests. 1 Enable Core1 monitoring for global writes.
19 M0WMEN	Master 0 Write Monitor Enable This bit enables monitoring of Core0 for global writes. 0 Disable Core0 monitoring. Global writes from Core0 do not initiate any snoop requests. 1 Enable Core0 monitoring for global writes.
20–21 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
22–24 CP1DS	Core1 Disable Snoop This field allows the suspension of snooping activity targeting Core1. Snooping can be disabled universally for Core1, upon detection of a Core1 snoop termination with error or queue overflow occurrence, or upon detection of an overflow of the Core0 snoop queue. 000 Core1 snooping disabled upon detection of an overflow of the Core1 snoop queue, as flagged in PCU_CESR[CP1ERR]. When PCU_CESR[CP1ERR] is cleared, Core1 snooping activity can resume. 001 Core1 snooping disabled upon detection of an overflow of the Core1 snoop queue or upon a Core1 snoop termination with error, as flagged in PCU_CESR[CP1ERR]. Any valid entries in the Core1 snoop queue will be invalidated and will not be processed. When PCU_CESR[CP1ERR] is cleared, Core1 snooping activity can resume. 010 Core1 snooping disabled upon detection of an overflow of the Core1 snoop queue or upon detection of an overflow of the Core0 snoop queue, as flagged in PCU_CESR[CP1ERR] or PCU_CESR[CP0ERR]. Any valid entries in the Core1 snoop queue will be invalidated and will not be processed. When PCU_CESR[CP1ERR] or PCU_CESR[CP0ERR] (as applicable) is cleared, Core1 snooping activity can resume. 011 Core1 snooping disabled upon a Core1 snoop termination with error or upon detection of an overflow of either the Core1 snoop queue or the Core0 snoop queue, as flagged in PCU_CESR[CP1ERR] or PCU_CESR[CP0ERR]. Any valid entries in the Core1 snoop queue will be invalidated and will not be processed. When PCU_CESR[CP1ERR] or PCU_CESR[CP0ERR] (as applicable) is cleared, Core1 snooping activity can resume. 1xx Core1 snooping entirely disabled. Any valid entries in the Core1 snoop queue will be invalidated and will not be processed.
25–27 CP0DS	Core0 Disable Snoop This field allows the suspension of snooping activity targeting Core0. Snooping can be disabled universally for Core0, upon detection of a Core0 snoop termination with error or queue overflow occurrence, or upon detection of an overflow of the Core1 snoop queue. 000 Core0 snooping disabled upon detection of an overflow of the Core0 snoop queue, as flagged in PCU_CESR[CP0ERR]. When PCU_CESR[CP0ERR] is cleared, Core0 snooping activity can resume. 001 Core0 snooping disabled upon detection of an overflow of the Core0 snoop queue or upon a Core0 snoop termination with error, as flagged in PCU_CESR[CP0ERR]. Any valid entries in the Core0 snoop queue will be invalidated and will not be processed. When PCU_CESR[CP0ERR] is cleared, Core0 snooping activity can resume. 010 Core0 snooping disabled upon detection of an overflow of the Core0 snoop queue or upon detection of an overflow of the Core1 snoop queue, as flagged in PCU_CESR[CP0ERR] or PCU_CESR[CP1ERR]. Any valid entries in the Core0 snoop queue will be invalidated and will not be processed. When PCU_CESR[CP0ERR] or PCU_CESR[CP1ERR] (as applicable) is cleared, Core0 snooping activity can resume.

Table continues on the next page...

PCU_CESR field descriptions (continued)

Field	Description
011	Core0 snooping disabled upon a Core0 snoop termination with error or upon detection of an overflow of either the Core0 snoop queue or the Core1 snoop queue, as flagged in PCU_CESR[CP0ERR] or PCU_CESR[CP1ERR]. Any valid entries in the Core0 snoop queue will be invalidated and will not be processed. When PCU_CESR[CP0ERR] or PCU_CESR[CP1ERR] (as applicable) is cleared, Core0 snooping activity can resume.
1xx	Core0 snooping entirely disabled. Any valid entries in the Core0 snoop queue will be invalidated and will not be processed.
28–29 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
30 SRST_EN	Software Reset Enable Writing 1 to this bit invokes a software-driven reset of the PCU. On the next cycle following the programming of PCU_CESR[SRST_EN], all programming model registers in the PCU are reset, and all snoop queue entries are invalidated. Then PCU_CESR[SRST_FLAG] is set to 1 to indicate that a software-driven reset of the PCU has occurred. Writing 0 to this bit has no effect. Reading this bit returns 0. 0 No software-driven reset is requested. Writing 0 to this field has no effect. Reading this bit always returns 0. 1 Invoke software-driven reset.
31 ENB	PCU Enable This bit provides a global enable/disable for the PCU. While the PCU is disabled, hardware coherency checking is not performed. 0 The PCU is disabled. 1 The PCU is enabled.

7.2.2 PCU Error Address Register, Core n (PCU_EAR_n)

This register can be read in all modes.

Address: 0h base + 10h offset + (16d × i), where i=0d to 1d

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	EADDR																															
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	

PCU_EAR_n field descriptions

Field	Description
0–31 EADDR	Error Address This read-only field is the reference address associated with the global write that resulted in a snoop error or overflow event.

7.2.3 PCU Error Detail Register, Core n (PCU_EDRn)

This register can be read in all modes.

Address: 0h base + 14h offset + (16d × i), where i=0d to 1d

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	EMSTR				EPTR				ESTAMP								0		EPROT				ERESP				0					
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	

PCU_EDRn field descriptions

Field	Description
0–3 EMSTR	Error Master This read-only field holds the AHB master ID associated with the global write that resulted in a snoop error or overflow event.
4–7 EPTR	Error Pointer This read-only field holds the value of the queue pointer associated with the global write that resulted in a snoop error or overflow event. Upon a snoop error event, this field describes the relative position of the corresponding entry in the 16-entry queue for the global write that resulted in a snoop error. Upon a queue overflow event, this field describes the relative position of the corresponding entry in the 16-entry queue for the global write that resulted in an overflow event.
8–15 ESTAMP	Error Time Stamp This read-only field holds the time stamp value associated with the snoop error or overflow event. The time stamp value is based on a free-running 8-bit timer that begins counting upon assertion of PCU_CESR[ENB]. Upon a snoop error event, this field captures the value of the time stamp at the time the snoop request was loaded into the queue. Upon a queue overflow event, this field captures the value of the time stamp at the time the overflow condition is detected.
16–17 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
18–23 EPROT	Error Protection Attribute This read-only field holds the AHB protection attributes that resulted in a snoop error or overflow event.
24–28 ERESP	Error Snoop Response This read-only field holds the snoop response associated with the snoop that terminated with error. This field is cleared when PCU_EDRn is loaded on an overflow event. Upon a queue overflow event, this field reads as all zeroes. In the field-setting descriptions, cc designates the number of collapsed requests: 00: no collapsing 01: two requests combined 10: three requests combined 11: four requests combined

Table continues on the next page...

PCU_EDR_n field descriptions (continued)

Field	Description
000cc	NULL: no operation performed or no matching cache entry
001cc	AutoInv: AutoInvalidation performed on clean unlocked lines with tag parity errors
010cc	ERROR: Error in processing a snoop request due to tag parity error
011cc	SYNC: Sync completed, snoop queue synchronized
100cc	Hit Clean: matching unlocked clean cache entry found
101cc	Hit Dirty: matching unlocked dirty cache entry found
110cc	Hit Locked: matching locked clean cache entry found
111cc	Hit Dirty Locked: matching locked clean cache entry found
29–31 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

7.2.4 PCU Interrupt Register, Core n (PCU_IR_n)

Access by mode:

- Read: all modes
- Write: Supervisor mode

Address: 0h base + 30h offset + (4d × i), where i=0d to 1d

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0														OIF	EIF
W															w1c	w1c
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

PCU_IR_n field descriptions

Field	Description
0–29 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
30 OIF	Overflow Interrupt Flag This field indicates an outstanding interrupt request as a result of a queue overflow. The flag and interrupt are cleared by writing 1 to this bit. Writing 0 to this bit has no effect. 0 No interrupt request 1 Interrupt request due to snoop termination with error
31 EIF	Error Interrupt Flag

Table continues on the next page...

PCU_IRn field descriptions (continued)

Field	Description
	This field indicates an outstanding interrupt request as a result of a snoop that terminated with error if PCU_CESR[CPnIEN]. The flag and interrupt are cleared by writing 1 to this bit. Writing 0 to this bit has no effect.
0	No interrupt request
1	Interrupt request due to snoop termination with error

Chapter 8

System Integration Unit (SIU, SIU_B)

8.1 Introduction

This chapter describes the device system integration units (SIU, and SIU_B) that configure and initialize the following features in the SIU:

- MCU reset configuration
- System reset operation
- Pad configuration
- External interrupts
- General-purpose I/O (GPIO)
- Internal peripheral multiplexing
- GPDI and GPDO I/Os of the DSPI modules
- TLB Entry Selector

as well as the following feature in SIU_B:

- Mirrored pad configuration and GPIO registers for Protected Port Output mode

8.1.1 Block diagram

The figure below is a block diagram of the SIU. The Power-on Reset Detection, Pad Ring/Pad Interface, and the Peripheral I/O Channels are external to the SIU.

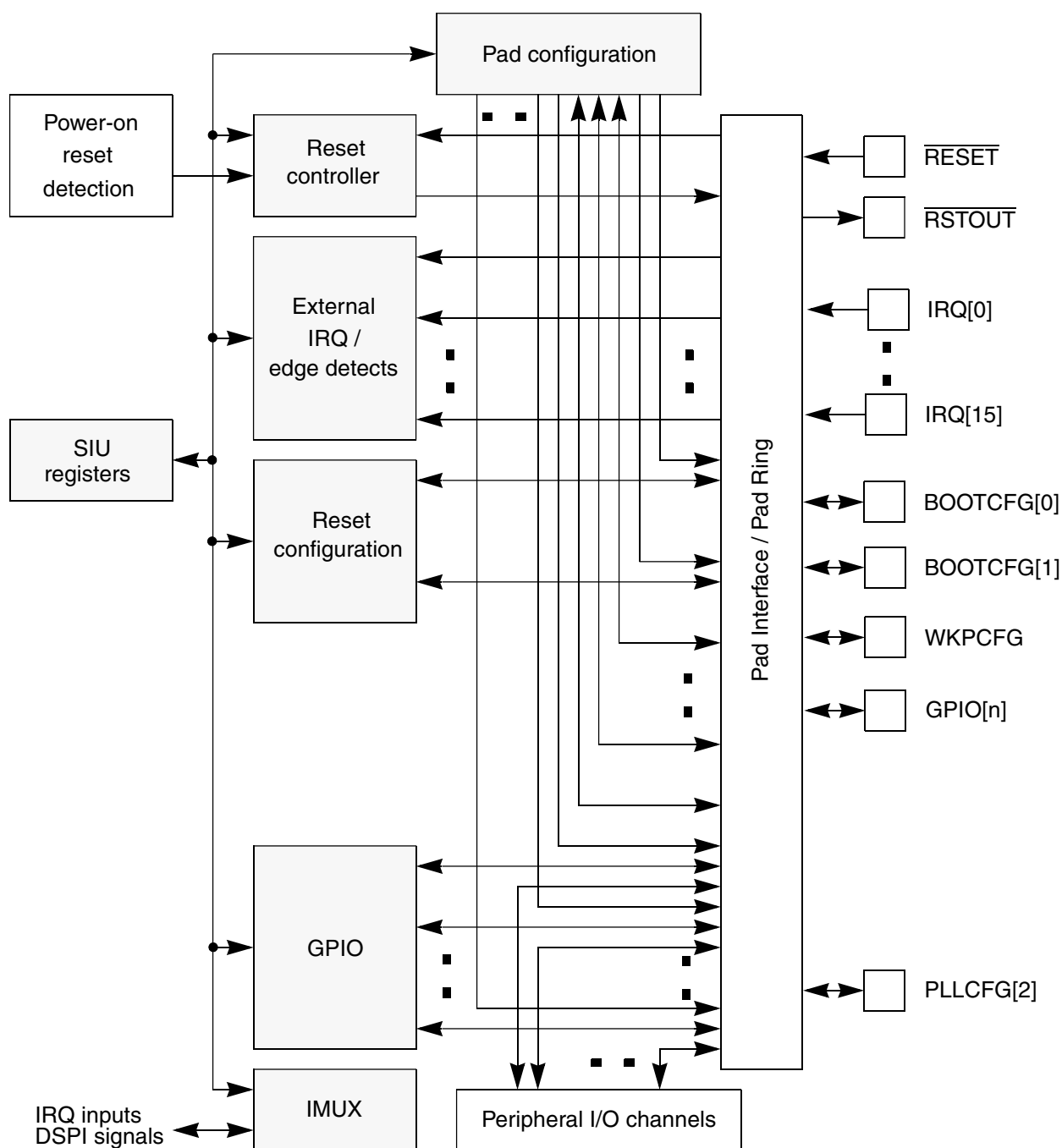


Figure 8-1. SIU block diagram

8.1.2 Overview

The following table lists the features that the SIU configures.

Table 8-1. SIU features

Feature	Description
MCU reset operations	Controls the external pin boot logic
System reset operations	Monitors internal and external reset sources, and drives the $\overline{\text{RSTOUT}}$ signal • Power-on reset support • Reset status register providing last reset source to software • Glitch detection on reset input • Software controlled reset assertion
Pad configuration registers	Enables the configuration and initialization of the I/O pin electrical characteristics using software to select the following: • Active function from the set of multiplexed functions • Pullup and pulldown characteristics of the pin • Slew rate for slow and medium pads • Open drain mode for output pins • Hysteresis for input pins • Drive strength of bus signals for fast pads
External interrupt operations	16 interrupt requests Rising- or falling-edge event detection Programmable digital filter for glitch rejection NMI and critical interrupt control
General purpose I/O (GPIO)	Provides uniform and discrete I/O control of MCU general-purpose I/O pins, where each GPIO signal has an input register and an output register.
Internal peripheral multiplexing	Provides flexibility to customize signal/pin assignments for application development that allows: • Flexible selection of eQADC trigger inputs • Assignment of interrupt requests (IRQs) between external pins and DSPI
TLB entry selector	Allows an external tool to non-intrusively change the TLB entry that is used by the cores to access memory.
Protected Port Output	Allows up to four GPIO pins to be reserved by any one of the two cores.

8.1.3 Modes of operation

8.1.3.1 Normal mode

In normal mode, the SIU provides the register interface and logic that controls the device and system configuration, the reset controller, and GPIO.

8.1.3.2 Debug mode

SIU operation in debug mode is identical to operation in normal mode.

8.2 Memory Map and Register Definition

Register offsets are relative to SIU's base address.

Pad Configuration Register (PCR) details

[Pad Configuration Register \(SIU_PCR \$n\$ \)](#) provides a general description of the PCRs. The following tables provide details about individual PCRs. Gaps exist in the memory map where registers correspond to GPIO pins that are not implemented in the package.

[Table 8-2](#) defines the symbols for PCR n fields' reset values that appear in [Table 8-3](#).

Table 8-2. SIU_PCR n field reset-value symbols

Symbol	Meaning
—	Not implemented in the PCR. Software should write a default value of 0.
0 or 1	Function is implemented. For its definition, see the applicable field's description.
U	Function is implemented and is defined by the WKPCFG bit of the Miscellaneous DCF client (or, in legacy mode, by the value of the WKPCFG pin at reset).

Table 8-3. SIU_PCR n settings

P C R	Offset (hex)	GPIO	Primary function	Alternate 1	Alternate 2	Alternate 3	SIU_PCR n [3:15] (SIU_PCR n [0:2] = reserved)												
							3	4	5	6	7	8	9	10	11	12	13	14	15
							P A 2	P A 1	P A 0	O B E	I B E	D S C 1	D S C 0	O D Y E S	H Y R C 1	S R P C 0	S R P C 1	W P P E S	W P P E S
75 ¹	00D6	GPIO75	MDO4				—	—	—	0	0	—	—	0	0	1	1	1	0
76 ¹	00D8	GPIO76	MDO5				—	—	—	0	0	—	—	0	0	1	1	1	0
77 ¹	00DA	GPIO77	MDO6				—	—	—	0	0	—	—	0	0	1	1	1	0
78 ¹	00DC	GPIO78	MDO7				—	—	—	0	0	—	—	0	0	1	1	1	0
79 ¹	00DE	GPIO79	MDO8				—	—	—	0	0	—	—	0	0	1	1	1	0
80 ¹	00E0	GPIO80	MDO9				—	—	—	0	0	—	—	0	0	1	1	1	0
81 ¹	00E2	GPIO81	MDO10				—	—	—	0	0	—	—	0	0	1	1	1	0
82 ¹	00E4	GPIO82	MDO11				—	—	—	0	0	—	—	0	0	1	1	1	0
83	00E6	GPIO83	CNTXA	TXDA ²	MCANTX0		—	0	0	0	0	—	—	0	0	0	0	1	1
84	00E8	GPIO84	CNRXA	RXDA ²	MCANRX0 ²		—	0	0	0	0	—	—	0	0	0	0	1	1
85	00EA	GPIO85	CNTXB	PCSC3			—	0	0	0	0	—	—	0	0	0	0	1	1
86	00EC	GPIO86	CNRXB	PCSC4			—	0	0	0	0	—	—	0	0	0	0	1	1
87	00EE	GPIO87	CNTXC	PCSD3			—	0	0	0	0	—	—	0	0	0	0	1	1

Table continues on the next page...

Table 8-3. SIU_PCRn settings
(continued)

P C R	Offset (hex)	GPIO	Primary function	Alternate 1	Alternate 2	Alternate 3	SIU_PCRn[3:15] (SIU_PCRn[0:2] = reserved)												
							3	4	5	6	7	8	9	10	11	12	13	14	15
							P A 2	P A 1	P A 0	O B E	I B E	D S C 1	D S C 0	O D S	H Y S	S R C 1	S R C 0	W P E	W P S
88	00F0	GPIO88	CNRXC	PCSD4	RXDE		—	0	0	0	0	—	—	0	0	0	0	1	1
89	00F2	GPIO89	TXDA ²				—	—	0	0	0	—	—	0	0	0	0	1	1
90	00F4	GPIO90	RXDA ²				—	—	0	0	0	—	—	0	0	0	0	1	1
91	00F6	GPIO91	TXDB ²	PCSD1	MCANRX0		—	0	0	0	0	—	—	0	0	0	0	1	1
92	00F8	GPIO92	RXDB ²	PCSD5			—	0	0	0	0	—	—	0	0	0	0	1	1
93	00FA	GPIO93	SCKA	PCSC1	MCANTX0	SENT1_B	0	0	0	0	0	—	—	0	0	0	0	1	1
94	00FC	GPIO94	SINA	PCSC2			—	0	0	0	0	—	—	0	0	0	0	1	1
95	00FE	GPIO95	SOUTA ²	PCSC5			—	0	0	0	0	—	—	0	0	0	0	1	1
96	0100	GPIO96	PCSA0	PCSD2			—	0	0	0	0	—	—	0	0	0	0	1	1
97	0102	GPIO97	PCSA1 ²	PCSE0			—	0	0	0	0	—	—	0	0	0	0	1	1
98	0104	GPIO98	PCSA2	SOUTE	FEC_TXD2	FEC_RXER	0	0	0	0	0	—	—	0	0	0	0	1	1
99	0106	GPIO99	PCSA3	SINE	FEC_MDIO	TXDD	0	0	0	0	0	—	—	0	0	0	0	1	1
100	0108	GPIO100	PCSA4 ²	SCKE	TXDE	FEC_RXCLK	0	0	0	0	0	—	—	0	0	0	0	1	1
101	010A	GPIO101	PCSA5	ETRIG1	RXDD	FEC_TXD3	0	0	0	0	0	—	—	0	0	0	0	1	1
102	010C	GPIO102	SCKB				—	—	0	0	0	—	—	0	0	0	0	1	1
103	010E	GPIO103	SINB				—	—	0	0	0	—	—	0	0	0	0	1	1
104	0110	GPIO104	SOUTB				—	—	0	0	0	—	—	0	0	0	0	1	1
105	0112	GPIO105	PCSB0 ²	PCSD2			—	0	0	0	0	—	—	0	0	0	0	1	1
106	0114	GPIO106	PCSB1 ²	PCSD0			—	0	0	0	0	—	—	0	0	0	0	1	1
107	0116	GPIO107	PCSB2	SOUTC	SENT2_B	FEC_RXD3	0	0	0	0	0	—	—	0	0	0	0	1	1
108	0118	GPIO108	PCSB3	SINC ²	SENT3_B		—	0	0	0	0	—	—	0	0	0	0	1	1
109	011A	GPIO109	PCSB4	SCKC ²	SENT4_B	FEC_MDC	0	0	0	0	0	—	—	0	0	0	0	1	1
110	011C	GPIO110	PCSB5	PCSC0 ²	SENT5_B	FEC_RXD2	0	0	0	0	0	—	—	0	0	0	0	1	1
113	0122	GPIO113	TCRCLKA	IRQ7			—	0	0	0	0	—	—	0	0	0	0	1	1
114	0124	GPIO114	ETPUA0	ETPUA12			—	0	0	0	0	—	—	0	0	0	0	1	U
115	0126	GPIO115	ETPUA1	ETPUA13			—	0	0	0	0	—	—	0	0	0	0	1	U
116	0128	GPIO116	ETPUA2	ETPUA14	RCH0_A		—	0	0	0	0	—	—	0	0	0	0	1	U
117	012A	GPIO117	ETPUA3	ETPUA15	RCH1_A	ERROR0	0	0	0	0	1	—	—	0	0	0	0	0	U
118	012C	GPIO118	ETPUA4	ETPUA16	RCH2_A		—	0	0	0	0	—	—	0	0	0	0	1	U
119	012E	GPIO119	ETPUA5	ETPUA17	RCH3_A		—	0	0	0	0	—	—	0	0	0	0	1	U
120	0130	GPIO120	ETPUA6	ETPUA18	RCH4_A		—	0	0	0	0	—	—	0	0	0	0	1	U
121	0132	GPIO121	ETPUA7	ETPUA19	RCH5_A		—	0	0	0	0	—	—	0	0	0	0	1	U

Table continues on the next page...

**Table 8-3. SIU_PCRn settings
(continued)**

P C R	Offset (hex)	GPIO	Primary function	Alternate 1	Alternate 2	Alternate 3	SIU_PCRn[3:15] (SIU_PCRn[0:2] = reserved)													
							3	4	5	6	7	8	9	10	11	12	13	14	15	
							P A 2	P A 1	P A 0	O B E	I B E	D S C 1	D S C 0	O D S	H Y S	S R C 1	S R C 0	W P E	W P S	
122	0134	GPIO122	ETPUA8	ETPUA20	RCH6_A		—	0	0	0	0	—	—	0	0	0	0	1	U	
123	0136	GPIO123	ETPUA9	ETPUA21	RCH7_A		—	0	0	0	0	—	—	0	0	0	0	1	U	
124	0138	GPIO124	ETPUA10	ETPUA22	RCH8_A	ERRORIN	0	0	0	0	1	—	—	0	0	0	0	1	U	
125	013A	GPIO125	ETPUA11	ETPUA23	RCH9_A	ERROR1	0	0	0	0	1	—	—	0	0	0	0	0	U	
126	013C	GPIO126	ETPUA12	PCSB1			—	0	0	0	0	—	—	0	0	0	0	1	U	
127	013E	GPIO127	ETPUA13	PCSB3			—	0	0	0	0	—	—	0	0	0	0	1	U	
128	0140	GPIO128	ETPUA14	PCSB4			—	0	0	0	0	—	—	0	0	0	0	1	U	
129	0142	GPIO129	ETPUA15	PCSB5	MCANTX0		—	0	0	0	0	—	—	0	0	0	0	1	U	
130	0144	GPIO130	ETPUA16	PCSD1			—	0	0	0	0	—	—	0	0	0	0	1	U	
131	0146	GPIO131	ETPUA17	PCSD2			—	0	0	0	0	—	—	0	0	0	0	1	U	
132	0148	GPIO132	ETPUA18	PCSD3			—	0	0	0	0	—	—	0	0	0	0	1	U	
133	014A	GPIO133	ETPUA19	PCSD4	MCANRX0 ²		—	0	0	0	0	—	—	0	0	0	0	1	U	
134	014C	GPIO134	ETPUA20	IRQ8			—	0	0	0	0	—	—	0	0	0	0	1	U	
135	014E	GPIO135	ETPUA21	IRQ9			—	0	0	0	0	—	—	0	0	0	0	1	U	
136	0150	GPIO136	ETPUA22	IRQ10	RCH0_A		—	0	0	0	0	—	—	0	0	0	0	1	U	
137	0152	GPIO137	ETPUA23	IRQ11	RCH1_A		—	0	0	0	0	—	—	0	0	0	0	1	U	
138	0154	GPIO138	ETPUA24	IRQ12	RCH2_A		—	0	0	0	0	—	—	0	0	0	0	1	U	
139	0156	GPIO139	ETPUA25	IRQ13	RCH3_A		—	0	0	0	0	—	—	0	0	0	0	1	U	
140	0158	GPIO140	ETPUA26	IRQ14	RCH4_A		—	0	0	0	0	—	—	0	0	0	0	1	U	
141	015A	GPIO141	ETPUA27	IRQ15	RCH5_A		—	0	0	0	0	—	—	0	0	0	0	1	U	
142	015C	GPIO142	ETPUA28	PCSC1	RCH6_A		—	0	0	0	0	—	—	0	0	0	0	1	U	
143	015E	GPIO143	ETPUA29	PCSC2	RCH7_A		—	0	0	0	0	—	—	0	0	0	0	1	U	
144	0160	GPIO144	ETPUA30	PCSC3	RCH8_A		—	0	0	0	0	—	—	0	0	0	0	1	U	
145	0162	GPIO145	ETPUA31	PCSC4	RCH9_A		—	0	0	0	0	—	—	0	0	0	0	1	U	
146	0164	GPIO146	TCRCLKB ²	IRQ6			—	0	0	0	0	—	—	0	0	0	0	1	1	
147	0166	GPIO147	ETPUB0	ETPUB16			—	0	0	0	0	—	—	0	0	0	0	1	U	
148	0168	GPIO148	ETPUB1	ETPUB17			—	0	0	0	0	—	—	0	0	0	0	1	U	
149	016A	GPIO149	ETPUB2	ETPUB18			—	0	0	0	0	—	—	0	0	0	0	1	U	
150	016C	GPIO150	ETPUB3	ETPUB19			—	0	0	0	0	—	—	0	0	0	0	1	U	
151	016E	GPIO151	ETPUB4	ETPUB20			—	0	0	0	0	—	—	0	0	0	0	1	U	
152	0170	GPIO152	ETPUB5	ETPUB21			—	0	0	0	0	—	—	0	0	0	0	1	U	
153	0172	GPIO153	ETPUB6	ETPUB22			—	0	0	0	0	—	—	0	0	0	0	1	U	

Table continues on the next page...

**Table 8-3. SIU_PCRn settings
(continued)**

P C R	Offset (hex)	GPIO	Primary function	Alternate 1	Alternate 2	Alternate 3	SIU_PCRn[3:15] (SIU_PCRn[0:2] = reserved)													
							3	4	5	6	7	8	9	10	11	12	13	14	15	
							P A 2	P A 1	P A 0	O B E	I B E	D S C 1	D S C 0	O D S	H Y S	S R C 1	S R C 0	W P E	W P S	
154	0174	GPIO154	ETPUB7	ETPUB23			—	0	0	0	0	—	—	0	0	0	0	1	U	
155	0176	GPIO155	ETPUB8	ETPUB24			—	0	0	0	0	—	—	0	0	0	0	1	U	
156	0178	GPIO156	ETPUB9	ETPUB25			—	0	0	0	0	—	—	0	0	0	0	1	U	
157	017A	GPIO157	ETPUB10	ETPUB26			—	0	0	0	0	—	—	0	0	0	0	1	U	
158	017C	GPIO158	ETPUB11	ETPUB27			—	0	0	0	0	—	—	0	0	0	0	1	U	
159	017E	GPIO159	ETPUB12	ETPUB28			—	0	0	0	0	—	—	0	0	0	0	1	U	
160	0180	GPIO160	ETPUB13	ETPUB29			—	0	0	0	0	—	—	0	0	0	0	1	U	
161	0182	GPIO161	ETPUB14	ETPUB30			—	0	0	0	0	—	—	0	0	0	0	1	U	
162	0184	GPIO162	ETPUB15	ETPUB31			—	0	0	0	0	—	—	0	0	0	0	1	U	
163	0186	GPIO163	ETPUB16	PCSA1			—	0	0	0	0	—	—	0	0	0	0	1	U	
164	0188	GPIO164	ETPUB17	PCSA2			—	0	0	0	0	—	—	0	0	0	0	1	U	
165	018A	GPIO165	ETPUB18	PCSA3			—	0	0	0	0	—	—	0	0	0	0	1	U	
166	018C	GPIO166	ETPUB19	PCSA4			—	0	0	0	0	—	—	0	0	0	0	1	U	
167	018E	GPIO167	ETPUB20				—	—	0	0	0	—	—	0	0	0	0	1	U	
168	0190	GPIO168	ETPUB21				—	—	0	0	0	—	—	0	0	0	0	1	U	
169	0192	GPIO169	ETPUB22	RCH0_B			—	0	0	0	0	—	—	0	0	0	0	1	U	
170	0194	GPIO170	ETPUB23	RCH1_B			—	0	0	0	0	—	—	0	0	0	0	1	U	
171	0196	GPIO171	ETPUB24	RCH2_B			—	0	0	0	0	—	—	0	0	0	0	1	U	
172	0198	GPIO172	ETPUB25	RCH3_B			—	0	0	0	0	—	—	0	0	0	0	1	U	
173	019A	GPIO173	ETPUB26	RCH4_B			—	0	0	0	0	—	—	0	0	0	0	1	U	
174	019C	GPIO174	ETPUB27	RCH5_B			—	0	0	0	0	—	—	0	0	0	0	1	U	
175	019E	GPIO175	ETPUB28	RCH6_B			—	0	0	0	0	—	—	0	0	0	0	1	U	
176	01A0	GPIO176	ETPUB29	RCH7_B			—	0	0	0	0	—	—	0	0	0	0	1	U	
177	01A2	GPIO177	ETPUB30	RCH8_B			—	0	0	0	0	—	—	0	0	0	0	1	U	
178	01A4	GPIO178	ETPUB31	RCH9_B			—	0	0	0	0	—	—	0	0	0	0	1	U	
179	01A6	GPIO179	EMIOS0	ETPUA0	RCH0_C		—	0	0	0	0	—	—	0	0	0	0	1	U	
180	01A8	GPIO180	EMIOS1	ETPUA1	RCH1_C	PCSE2	0	0	0	0	0	—	—	0	0	0	0	1	U	
181	01AA	GPIO181	EMIOS2	ETPUA2	RCH2_C	PCSE3	0	0	0	0	0	—	—	0	0	0	0	1	U	
182	01AC	GPIO182	EMIOS3	ETPUA3	RCH3_C	PCSE4	0	0	0	0	0	—	—	0	0	0	0	1	U	
183	01AE	GPIO183	EMIOS4	ETPUA4	RCH4_C	PCSE5	0	0	0	0	0	—	—	0	0	0	0	1	U	
184	01B0	GPIO184	EMIOS5	ETPUA5	RCH5_C		—	0	0	0	0	—	—	0	0	0	0	1	U	
185	01B2	GPIO185	EMIOS6	ETPUA6	RCH6_C		—	0	0	0	0	—	—	0	0	0	0	1	U	

Table continues on the next page...

**Table 8-3. SIU_PCRn settings
(continued)**

P C R	Offset (hex)	GPIO	Primary function	Alternate 1	Alternate 2	Alternate 3	SIU_PCRn[3:15] (SIU_PCRn[0:2] = reserved)													
							3	4	5	6	7	8	9	10	11	12	13	14	15	
							P A 2	P A 1	P A 0	O B E	I B E	D S C 1	D S C 0	O D E	H Y S	S R C 1	S R C 0	W P E	W P S	
186	01B4	GPIO186	EMIOS7	ETPUA7	RCH7_C		—	0	0	0	0	—	—	0	0	0	0	1	U	
187	01B6	GPIO187	EMIOS8	ETPUA8	RCH8_C		—	0	0	0	0	—	—	0	0	0	0	1	U	
188	01B8	GPIO188	EMIOS9	ETPUA9	RCH9_C		—	0	0	0	0	—	—	0	0	0	0	1	U	
189	01BA	GPIO189	EMIOS10	SCKD			—	0	0	0	0	—	—	0	0	0	0	1	U	
190	01BC	GPIO190	EMIOS11	SIND			—	0	0	0	0	—	—	0	0	0	0	1	U	
191	01BE	GPIO191	EMIOS12	SOUTC			—	0	0	0	0	—	—	0	0	0	0	1	U	
192	01C0	GPIO192	EMIOS13	SOUTD			—	0	0	0	0	—	—	0	0	0	0	1	U	
193	01C2	GPIO193	EMIOS14	IRQ0	CNTXD	RCH0_C	0	0	0	0	0	—	—	0	0	0	0	1	U	
194	01C4	GPIO194	EMIOS15	IRQ1	CNRXD	RCH1_C	0	0	0	0	0	—	—	0	0	0	0	1	U	
195	01C6	GPIO195	EMIOS16	ETPUB0	MCANTX1	RCH2_C	0	0	0	0	0	—	—	0	0	0	0	1	U	
196	01C8	GPIO196	EMIOS17	ETPUB1	MCANRX1	RCH3_C	0	0	0	0	0	—	—	0	0	0	0	1	U	
197	01CA	GPIO197	EMIOS18	ETPUB2	RCH4_C		—	0	0	0	0	—	—	0	0	0	0	1	U	
198	01CC	GPIO198	EMIOS19	ETPUB3	RCH5_C		—	0	0	0	0	—	—	0	0	0	0	1	U	
199	01CE	GPIO199	EMIOS20	ETPUB4	RCH6_C		—	0	0	0	0	—	—	0	0	0	0	1	U	
200	01D0	GPIO200	EMIOS21	ETPUB5	RCH7_C		—	0	0	0	0	—	—	0	0	0	0	1	U	
201	01D2	GPIO201	EMIOS22	ETPUB6	RCH8_C		—	0	0	0	0	—	—	0	0	0	0	1	U	
202	01D4	GPIO202	EMIOS23	ETPUB7	RCH9_C		—	0	0	0	0	—	—	0	0	0	0	1	U	
203	01D6	GPIO203	EMIOS24	PCSB0 ²			—	0	0	0	0	—	—	0	0	0	0	1	U	
204	01D8	GPIO204	EMIOS25	PCSB1			—	0	0	0	0	—	—	0	0	0	0	1	U	
208	01E0	GPIO208	PLLCFG0 ²	IRQ4	ETPUC9	SIPI_TXN	0	0	1	0	0	—	—	0	1	0	0	1	1	
209	01E2	GPIO209	PLLCFG1 ²	IRQ5	SOUTD	SIPI_TXP	0	0	1	0	0	—	—	0	1	0	0	1	1	
210	01E4	GPIO210	PLLCFG2	ETPUC3	PCSE1	SIPI_REFCLK	0	0	0	0	0	—	—	0	1	0	0	1	1	
211	01E6	GPIO211	BOOTCFG0 ²	IRQ2			—	0	1	0	0	—	—	0	1	0	0	1	0	
212	01E8	GPIO212	BOOTCFG1 ²	IRQ3	ETPUC0	SIPI_RXP	0	0	1	0	0	—	—	0	1	0	0	1	0	
213	01EA	GPIO213	WKPCFG	NMI ³	ETPUC1	SIPI_RXN	0	0	1	0	0	—	—	0	1	0	0	1	1	
214	01EC		ENGCLK				—	—	—	1	—	—	—	—	—	1	1	—	—	
219	01F6		MCKO				—	—	—	—	—	—	—	—	—	1	1	—	—	
220 ¹	01F8	GPIO220	MDO0				—	—	—	0	0	—	—	0	0	1	1	1	1	
221 ¹	01FA	GPIO221	MDO1				—	—	—	0	0	—	—	0	0	1	1	1	0	
222 ¹	01FC	GPIO222	MDO2				—	—	—	0	0	—	—	0	0	1	1	1	0	
223 ¹	01FE	GPIO223	MDO3	TXDE ²			—	0	0	0	0	—	—	0	0	1	1	1	0	
224	0200		MSEO0				—	—	—	—	—	—	—	—	—	1	1	—	—	

Table continues on the next page...

Table 8-3. SIU_PCRn settings
(continued)

P C R	Offset (hex)	GPIO	Primary function	Alternate 1	Alternate 2	Alternate 3	SIU_PCRn[3:15] (SIU_PCRn[0:2] = reserved)												
							3	4	5	6	7	8	9	10	11	12	13	14	15
							P A 2	P A 1	P A 0	O B E	I B E	D S C 1	D S C 0	O D S	H Y S	S R C 1	S R C 0	W P E	W P S
225	0202		MSEO1				—	—	—	—	—	—	—	—	—	1	1	—	—
226	0204		RDY				—	—	—	—	—	—	—	—	—	1	1	—	—
227	0206		EVTO				—	—	—	—	—	—	—	—	—	1	1	—	—
228	0208		TDO				—	—	—	—	—	—	—	—	—	1	1	—	—
229	020A		D_CLKOUT				—	—	—	1	—	1	1	—	—	—	—	—	—
230	020C		RSTOUT				—	—	—	—	—	—	—	—	—	1	1	—	—
231 ⁴	020E	GPIO231	MDO12				—	—	—	0	0	—	—	0	0	1	1	1	0
232 ¹	0210	GPIO232	MDO13				—	—	—	0	0	—	—	0	0	1	1	1	0
233 ¹	0212	GPIO233	MDO14				—	—	—	0	0	—	—	0	0	1	1	1	0
234 ¹	0214	GPIO234	MDO15				—	—	—	0	0	—	—	0	0	1	1	1	0
235	0216	GPIO235	SCKC ²	SCK_C_LVD S+			—	0	0	0	0	—	—	0	0	0	0	1	1
236	0218	GPIO236	SINC ²	SCK_C_LVD S-			—	0	0	0	0	—	—	0	0	0	0	1	1
237	021A	GPIO237	SOUTC	SOUT_C_LV DS+			—	0	0	0	0	—	—	0	0	0	0	1	1
238	021C	GPIO238	PCSC0 ²	SOUT_C_LV DS-			—	0	0	0	0	—	—	0	0	0	0	1	1
239	021E	GPIO239	PCSC1	TXDD			—	0	0	0	0	—	—	0	0	0	0	1	1
240	0220	GPIO240	PCSC2	SCK_D_LVD S+	SCKD		—	0	0	0	0	—	—	0	0	0	0	1	1
241	0222	GPIO241	PCSC3	SCK_D_LVD S-	SIND		—	0	0	0	0	—	—	0	0	0	0	1	1
242	0224	GPIO242	PCSC4	MCANTX0	SOUT_D_LV DS+	SOUTD	0	0	0	0	0	—	—	0	0	0	0	1	1
243	0226	GPIO243	PCSC5	MCANRX0	SOUT_D_LV DS-	PCSD0	0	0	0	0	0	—	—	0	0	0	0	1	1
244	0228	GPIO244	TXDC	ETRIG0			—	0	0	0	0	—	—	0	0	0	0	1	1
245	022A	GPIO245	RXDC				—	—	0	0	0	—	—	0	0	0	0	1	1
246	022C	GPIO246	CNTXD		MCANTX1		—	0	0	0	0	—	—	0	0	0	0	1	1
247	022E	GPIO247	CNRXD ²		MCANRX1		—	0	0	0	0	—	—	0	0	0	0	1	1
248	0230	GPIO248	FEC_TXD0 ²	SENT0_A			—	0	0	0	0	—	—	0	0	1	1	1	1
249	0232	GPIO249	FEC_RX_DV ²	SENT1_A			—	0	0	0	0	—	—	0	0	1	1	1	1
250	0234	GPIO250	FEC_RXD0 ²	SENT2_A			—	0	0	0	0	—	—	0	0	1	1	1	1

Table continues on the next page...

**Table 8-3. SIU_PCR_n settings
(continued)**

P C R	Offset (hex)	GPIO	Primary function	Alternate 1	Alternate 2	Alternate 3	SIU_PCRn[3:15] (SIU_PCRn[0:2] = reserved)													
							3	4	5	6	7	8	9	10	11	12	13	14	15	
							P A 2	P A 1	P A 0	O B E	I B E	D S C 1	D S C 0	O D E	H Y S	S R C 1	S R C 0	W P E	W P S	
251	0236	GPIO251	FEC_TXD1 ²	SENT3_A			—	0	0	0	0	—	—	0	0	1	1	1	1	
252	0238	GPIO252	FEC_TX_EN ²	SENT4_A			—	0	0	0	0	—	—	0	0	1	1	1	1	
253	023A	GPIO253	FEC_RXD1 ²	SENT5_A			—	0	0	0	0	—	—	0	0	1	1	1	1	
256	0240	GPIO256	D_CS0				—	—	0	0	0	1	1	0	0	—	—	1	1	
257	0242	GPIO257	D_CS2	D_ADD_DAT 31			—	0	0	0	0	1	1	0	0	—	—	1	1	
258	0244	GPIO258	D_CS3	D_TEA			—	0	0	0	0	1	1	0	0	—	—	1	1	
259	0246	GPIO259	D_ADD12				—	—	0	0	0	1	1	0	0	—	—	1	1	
260	0248	GPIO260	D_ADD13				—	—	0	0	0	1	1	0	0	—	—	1	1	
261	024A	GPIO261	D_ADD14				—	—	0	0	0	1	1	0	0	—	—	1	1	
262	024C	GPIO262	D_ADD15				—	—	0	0	0	1	1	0	0	—	—	1	1	
263	024E	GPIO263	D_ADD16	D_ADD_DAT 16			—	0	0	0	0	1	1	0	0	—	—	1	1	
264	0250	GPIO264	D_ADD17	D_ADD_DAT 17			—	0	0	0	0	1	1	0	0	—	—	1	1	
265	0252	GPIO265	D_ADD18	D_ADD_DAT 18			—	0	0	0	0	1	1	0	0	—	—	1	1	
266	0254	GPIO266	D_ADD19	D_ADD_DAT 19			—	0	0	0	0	1	1	0	0	—	—	1	1	
267	0256	GPIO267	D_ADD20	D_ADD_DAT 20			—	0	0	0	0	1	1	0	0	—	—	1	1	
268	0258	GPIO268	D_ADD21	D_ADD_DAT 21			—	0	0	0	0	1	1	0	0	—	—	1	1	
269	025A	GPIO269	D_ADD22	D_ADD_DAT 22			—	0	0	0	0	1	1	0	0	—	—	1	1	
270	025C	GPIO270	D_ADD23	D_ADD_DAT 23			—	0	0	0	0	1	1	0	0	—	—	1	1	
271	025E	GPIO271	D_ADD24	D_ADD_DAT 24			—	0	0	0	0	1	1	0	0	—	—	1	1	
272	0260	GPIO272	D_ADD25	D_ADD_DAT 25			—	0	0	0	0	1	1	0	0	—	—	1	1	
273	0262	GPIO273	D_ADD26	D_ADD_DAT 26			—	0	0	0	0	1	1	0	0	—	—	1	1	
274	0264	GPIO274	D_ADD27	D_ADD_DAT 27			—	0	0	0	0	1	1	0	0	—	—	1	1	
275	0266	GPIO275	D_ADD28	D_ADD_DAT 28			—	0	0	0	0	1	1	0	0	—	—	1	1	

Table continues on the next page...

**Table 8-3. SIU_PCRn settings
(continued)**

P C R	Offset (hex)	GPIO	Primary function	Alternate 1	Alternate 2	Alternate 3	SIU_PCRn[3:15] (SIU_PCRn[0:2] = reserved)													
							3	4	5	6	7	8	9	10	11	12	13	14	15	
							P A 2	P A 1	P A 0	O B E	I B E	D S C 1	D S C 0	O D E	H Y S	S R C 1	S R C 0	W P E	W P S	
276	0268	GPIO276	D_ADD29	D_ADD_DAT 29			—	0	0	0	0	1	1	0	0	—	—	1	1	
277	026A	GPIO277	D_ADD30	D_ADD_DAT 30			—	0	0	0	0	1	1	0	0	—	—	1	1	
278	026C	GPIO278	D_DAT0				—	—	0	0	0	—	—	0	0	1	1	1	1	
279	026E	GPIO279	D_DAT1				—	—	0	0	0	—	—	0	0	1	1	1	1	
280	0270	GPIO280	D_DAT2				—	—	0	0	0	—	—	0	0	1	1	1	1	
281	0272	GPIO281	D_DAT3				—	—	0	0	0	—	—	0	0	1	1	1	1	
282	0274	GPIO282	D_DAT4				—	—	0	0	0	—	—	0	0	1	1	1	1	
283	0276	GPIO283	D_DAT5				—	—	0	0	0	—	—	0	0	1	1	1	1	
284	0278	GPIO284	D_DAT6				—	—	0	0	0	—	—	0	0	1	1	1	1	
285	027A	GPIO285	D_DAT7				—	—	0	0	0	—	—	0	0	1	1	1	1	
286	027C	GPIO286	D_DAT8				—	—	0	0	0	—	—	0	0	1	1	1	1	
287	027E	GPIO287	D_DAT9				—	—	0	0	0	—	—	0	0	1	1	1	1	
288	0280	GPIO288	D_DAT10				—	—	0	0	0	—	—	0	0	1	1	1	1	
289	0282	GPIO289	D_DAT11				—	—	0	0	0	—	—	0	0	1	1	1	1	
290	0284	GPIO290	D_DAT12				—	—	0	0	0	—	—	0	0	1	1	1	1	
291	0286	GPIO291	D_DAT13				—	—	0	0	0	—	—	0	0	1	1	1	1	
292	0288	GPIO292	D_DAT14				—	—	0	0	0	—	—	0	0	1	1	1	1	
293	028A	GPIO293	D_DAT15				—	—	0	0	0	—	—	0	0	1	1	1	1	
294	028C	GPIO294	D_RD_WR				—	—	0	0	0	1	1	0	0	—	—	1	1	
295	028E	GPIO295	D_WE0				—	—	0	0	0	1	1	0	0	—	—	1	1	
296	0290	GPIO296	D_WE1				—	—	0	0	0	1	1	0	0	—	—	1	1	
297	0292	GPIO297	D_OE				—	—	0	0	0	1	1	0	0	—	—	1	1	
298	0294	GPIO298	D_TS				—	—	0	0	0	1	1	0	0	—	—	1	1	
299	0296	GPIO299	D_ALE				—	—	0	0	0	—	—	0	0	1	1	1	1	
300	0298	GPIO300	D_TA				—	—	0	0	0	1	1	0	0	—	—	1	1	
301	029A	GPIO301	D_CS1				—	—	0	0	0	1	1	0	0	—	—	1	1	
302	029C	GPIO302	D_BDIP				—	—	0	0	0	1	1	0	0	—	—	1	1	
303	029E	GPIO303	D_WE2				—	—	0	0	0	1	1	0	0	—	—	1	1	
304	02A0	GPIO304	D_WE3				—	—	0	0	0	1	1	0	0	—	—	1	1	
305	02A2	GPIO305	D_ADD9				—	—	0	0	0	1	1	0	0	—	—	1	1	
306	02A4	GPIO306	D_ADD10				—	—	0	0	0	1	1	0	0	—	—	1	1	

Table continues on the next page...

**Table 8-3. SIU_PCRn settings
(continued)**

P C R	Offset (hex)	GPIO	Primary function	Alternate 1	Alternate 2	Alternate 3	SIU_PCRn[3:15] (SIU_PCRn[0:2] = reserved)														
							3	4	5	6	7	8	9	10	11	12	13	14	15		
							P A 2	P A 1	P A 0	O B E	I B E	D S C 1	D S C 0	O D E	H Y S	S R C 1	S R C 0	W P E	W P S		
307	02A6	GPIO307	D_ADD11				—	—	0	0	0	1	1	0	0	—	—	1	1		
432	03A0	GPIO432	EMIOS26	PCSB2			—	0	0	0	0	—	—	0	0	0	0	1	U		
433	03A2	GPIO433	EMIOS27	PCSB3			—	0	0	0	0	—	—	0	0	0	0	1	U		
434	03A4	GPIO434	EMIOS28	PCSC0 ²			—	0	0	0	0	—	—	0	0	0	0	1	U		
435	03A6	GPIO435	EMIOS29	PCSC1			—	0	0	0	0	—	—	0	0	0	0	1	U		
436	03A8	GPIO436	EMIOS30	PCSC2			—	0	0	0	0	—	—	0	0	0	0	1	U		
437	03AA	GPIO437	EMIOS31	PCSC5			—	0	0	0	0	—	—	0	0	0	0	1	U		
440	03B0	GPI440	TCRCLKC	IRQ6 ²	ANB40		—	0	0	—	0	—	—	—	0	—	—	1	1		
441	03B2	GPI441	ETPUC0	SENT0_A	ANB41		—	0	0	—	0	—	—	—	0	—	—	1	U		
442	03B4	GPI442	ETPUC1	SENT1_A	ANB42		—	0	0	—	0	—	—	—	0	—	—	1	U		
443	03B6	GPIO443	ETPUC2	SENT3_A			—	0	0	0	0	—	—	0	0	0	0	1	U		
444	03B8	GPI444	ETPUC3	SENT4_A	ANB45		—	0	0	—	0	—	—	—	0	—	—	1	U		
445	03BA	GPIO445	ETPUC4	SENT5_A	TCRCLKC		—	0	0	0	0	—	—	0	0	0	0	1	U		
446	03BC	GPIO446	ETPUC5	PCSE2	SENT0_B		—	0	0	0	0	—	—	0	0	0	0	1	U		
447	03BE	GPIO447	ETPUC6	PCSE3	SENT1_B ²		—	0	0	0	0	—	—	0	0	0	0	1	U		
448	03C0	GPIO448	ETPUC7	PCSE4	SENT2_B ²		—	0	0	0	0	—	—	0	0	0	0	1	U		
449	03C2	GPIO449	ETPUC8 ²	PCSE5	SENT3_B ²		—	0	0	0	0	—	—	0	0	0	0	1	U		
450	03C4	GPI450	ETPUC9 ²	IRQ0 ²	SENT4_B ²	ANB44	—	0	0	—	0	—	—	—	0	—	—	1	U		
451	03C6	GPIO451	ETPUC10 ²	IRQ1 ²	SENT5_B ²		—	0	0	0	0	—	—	0	0	0	0	1	U		
452	03C8	GPIO452	ETPUC11 ²	IRQ2 ²			—	0	0	0	0	—	—	0	0	0	0	1	U		
453	03CA	GPIO453	ETPUC12 ²	IRQ3 ²	MCANTX1		—	0	0	0	0	—	—	0	0	0	0	1	U		
454	03CC	GPIO454	ETPUC13 ²	IRQ4 ²	SDOUT0_PSI5A	RCH0_C	0	0	0	0	0	—	—	0	0	0	0	1	U		
455	03CE	GPIO455	ETPUC14 ²	IRQ5 ²	SDIN0_PSI5A	RCH1_C	0	0	0	0	0	—	—	0	0	0	0	1	U		
456	03D0	GPIO456	ETPUC15 ²	MCANRX1 ²	RCH2_C		—	0	0	0	0	—	—	0	0	0	0	1	U		
457	03D2	GPIO457	ETPUC16	MCANTX1	RCH3_C		—	0	0	0	0	—	—	0	0	0	0	1	U		
458	03D4	GPIO458	ETPUC17	TXDF	RCH4_C		—	0	0	0	0	—	—	0	0	0	0	1	U		
459	03D6	GPIO459	ETPUC18	RXDF			—	0	0	0	0	—	—	0	0	0	0	1	U		
460	03D8	GPIO460	ETPUC19	TXDA	SDOUT0_PSI5B		—	0	0	0	0	—	—	0	0	0	0	1	U		
461	03DA	GPIO461	ETPUC20	RXDA	SDIN0_PSI5B		—	0	0	0	0	—	—	0	0	0	0	1	U		
462	03DC	GPIO462	ETPUC21	TXDB ²			—	0	0	0	0	—	—	0	0	0	0	1	U		

Table continues on the next page...

Table 8-3. SIU_PCRn settings
(continued)

P C R	Offset (hex)	GPIO	Primary function	Alternate 1	Alternate 2	Alternate 3	SIU_PCRn[3:15] (SIU_PCRn[0:2] = reserved)													
							3	4	5	6	7	8	9	10	11	12	13	14	15	
							P A 2	P A 1	P A 0	O B E	I B E	D S C 1	D S C 0	O D S	H Y S	S R C 1	S R C 0	W P E	W P S	
463	03DE	GPIO463	ETPUC22	RXDB ²			—	0	0	0	0	0	—	—	0	0	0	0	1	U
464	03E0	GPIO464	ETPUC23	PCSD5	MAA0	MAB0	0	0	0	0	0	—	—	0	0	0	0	0	1	U
465	03E2	GPIO465	ETPUC24	PCSD4	MAA1	MAB1	0	0	0	0	0	—	—	0	0	0	0	0	1	U
466	03E4	GPIO466	ETPUC25	PCSD3	MAA2	MAB2	0	0	0	0	0	—	—	0	0	0	0	0	1	U
467	03E6	GPIO467	ETPUC26	PCSD2	MAB0		—	0	0	0	0	—	—	0	0	0	0	0	1	U
468	03E8	GPIO468	ETPUC27	PCSD1	RCH5_C	MAB1	0	0	0	0	0	—	—	0	0	0	0	0	1	U
469	03EA	GPIO469	ETPUC28	PCSD0 ²	RCH6_C	MAB2	0	0	0	0	0	—	—	0	0	0	0	0	1	U
470	03EC	GPIO470	ETPUC29	SCKD ²	RCH7_C		—	0	0	0	0	—	—	0	0	0	0	0	1	U
471	03EE	GPIO471	ETPUC30	SOUTD	RCH8_C		—	0	0	0	0	—	—	0	0	0	0	0	1	U
472	03F0	GPIO472	ETPUC31	SIND ²	RCH9_C		—	0	0	0	0	—	—	0	0	0	0	0	1	U
474	03F4	GPI474	FEC_TXCLK _REFCLK	SENT0_B ²			—	0	0	—	0	—	—	—	0	—	—	—	1	1
475	03F6	GPI475	SENT2_A ²	ANB43			—	0	0	—	0	—	—	—	0	—	—	—	1	U
476	03F8	GPI476	ANB8	ETPUA31	EMIOS16		—	0	0	—	0	—	—	—	0	—	—	—	1	U
477	03FA	GPI477	ANB9	ETPUA30	EMIOS17		—	0	0	—	0	—	—	—	0	—	—	—	1	U
478	03FC	GPI478	ANB10	ETPUA29	EMIOS18		—	0	0	—	0	—	—	—	0	—	—	—	1	U
479	03FE	GPI479	ANB11	ETPUA28	EMIOS19		—	0	0	—	0	—	—	—	0	—	—	—	1	U
480	0400	GPI480	ANB12	ETPUA27	EMIOS20		—	0	0	—	0	—	—	—	0	—	—	—	1	U
481	0402	GPI481	ANB13	ETPUA26	EMIOS21		—	0	0	—	0	—	—	—	0	—	—	—	1	U
482	0404	GPI482	ANB14	ETPUA25	EMIOS22		—	0	0	—	0	—	—	—	0	—	—	—	1	U
483	0406	GPI483	ANB15	ETPUA24	EMIOS23		—	0	0	—	0	—	—	—	0	—	—	—	1	U
484	0408	GPI484	ANB16	ETPUC15 ²	EMIOS24		—	0	0	—	0	—	—	—	0	—	—	—	1	U
485	040A	GPI485	ANB17	ETPUC14 ²	EMIOS25		—	0	0	—	0	—	—	—	0	—	—	—	1	U
486	040C	GPI486	ANB18	ETPUC13 ²	EMIOS26		—	0	0	—	0	—	—	—	0	—	—	—	1	U
487	040E	GPI487	ANB19	ETPUC12 ²	EMIOS27		—	0	0	—	0	—	—	—	0	—	—	—	1	U
488	0410	GPI488	ANB20	ETPUC11 ²	EMIOS28		—	0	0	—	0	—	—	—	0	—	—	—	1	U
489	0412	GPI489	ANB21	ETPUC10 ²	EMIOS29		—	0	0	—	0	—	—	—	0	—	—	—	1	U
490	0414	GPI490	ANB22	ETPUC9 ²	EMIOS30		—	0	0	—	0	—	—	—	0	—	—	—	1	U
491	0416	GPI491	ANB23	ETPUC8 ²	EMIOS31		—	0	0	—	0	—	—	—	0	—	—	—	1	U

1. The choice between MDO and GPIO functions is based on Nexus programming. If Nexus is enabled, this pin takes the role of MDO, otherwise it is GPIO. The OBE and IBE fields only apply when the pin is working as GPIO, otherwise it is an output-only MDO pin.
2. This input function is also present in other pins, but only one pin can be selected at any given time. Select the pin to drive the function using the SIU_IMUX registers.

Memory Map and Register Definition

- NMI function is not selected by the PA settings. It is enabled by write-once bits in the SIU_IREER and SIU_IFEER registers. Once NMI is enabled by one of these registers, the PA settings are ignored.
- The choice between MDO and GPIO functions is based on Nexus programming. If Nexus is enabled and configured to use Full Port Mode, then this pin takes the role of MDO. If Nexus is not enabled or it is enabled but configured to use Reduced Port Mode, then this pin takes the role of GPIO. The OBE and IBE fields only apply when the pin is working as GPIO, otherwise it is an output-only MDO pin.

SIU memory map

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
4	MCU Identification Register (SIU_MIDR)	32	R	See section	8.2.1/236
C	Reset Status Register (SIU_RSR)	32	w1c	See section	8.2.2/237
10	System Reset Control Register (SIU_SRCR)	32	R/W	0000_0000h	8.2.3/240
14	External IRQ Status Register (SIU_EISR)	32	R	0000_0000h	8.2.4/241
18	DMA/Interrupt Request Enable Register (SIU_DIRER)	32	R/W	0000_0000h	8.2.5/244
1C	DMA/Interrupt Request Select Register (SIU_DIRSR)	32	R/W	0000_0000h	8.2.6/247
20	Overrun Status Register (SIU_OSR)	32	w1c	0000_0000h	8.2.7/249
24	Overrun Request Enable Register (SIU_ORER)	32	R/W	0000_0000h	8.2.8/251
28	IRQ Rising-Edge Event Enable Register (SIU_IREER)	32	R/W	0000_0000h	8.2.9/253
2C	IRQ Falling-Edge Event Enable Register (SIU_IFEER)	32	R/W	0000_0000h	8.2.10/255
30	IRQ Digital Filter Register (SIU_IDFR)	32	R/W	0000_0000h	8.2.11/258
34	IRQ Filtered Input Register (SIU_IFIR)	32	R	0000_0000h	8.2.12/258
D6	Pad Configuration Register (SIU_PCR75)	16	R/W	See section	8.2.13/260
D8	Pad Configuration Register (SIU_PCR76)	16	R/W	See section	8.2.13/260
DA	Pad Configuration Register (SIU_PCR77)	16	R/W	See section	8.2.13/260
DC	Pad Configuration Register (SIU_PCR78)	16	R/W	See section	8.2.13/260
DE	Pad Configuration Register (SIU_PCR79)	16	R/W	See section	8.2.13/260
E0	Pad Configuration Register (SIU_PCR80)	16	R/W	See section	8.2.13/260
E2	Pad Configuration Register (SIU_PCR81)	16	R/W	See section	8.2.13/260
E4	Pad Configuration Register (SIU_PCR82)	16	R/W	See section	8.2.13/260
E6	Pad Configuration Register (SIU_PCR83)	16	R/W	See section	8.2.13/260
E8	Pad Configuration Register (SIU_PCR84)	16	R/W	See section	8.2.13/260
EA	Pad Configuration Register (SIU_PCR85)	16	R/W	See section	8.2.13/260
EC	Pad Configuration Register (SIU_PCR86)	16	R/W	See section	8.2.13/260
EE	Pad Configuration Register (SIU_PCR87)	16	R/W	See section	8.2.13/260
F0	Pad Configuration Register (SIU_PCR88)	16	R/W	See section	8.2.13/260
F2	Pad Configuration Register (SIU_PCR89)	16	R/W	See section	8.2.13/260
F4	Pad Configuration Register (SIU_PCR90)	16	R/W	See section	8.2.13/260
F6	Pad Configuration Register (SIU_PCR91)	16	R/W	See section	8.2.13/260
F8	Pad Configuration Register (SIU_PCR92)	16	R/W	See section	8.2.13/260
FA	Pad Configuration Register (SIU_PCR93)	16	R/W	See section	8.2.13/260
FC	Pad Configuration Register (SIU_PCR94)	16	R/W	See section	8.2.13/260
FE	Pad Configuration Register (SIU_PCR95)	16	R/W	See section	8.2.13/260
100	Pad Configuration Register (SIU_PCR96)	16	R/W	See section	8.2.13/260

Table continues on the next page...

SIU memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
102	Pad Configuration Register (SIU_PCR97)	16	R/W	See section	8.2.13/260
104	Pad Configuration Register (SIU_PCR98)	16	R/W	See section	8.2.13/260
106	Pad Configuration Register (SIU_PCR99)	16	R/W	See section	8.2.13/260
108	Pad Configuration Register (SIU_PCR100)	16	R/W	See section	8.2.13/260
10A	Pad Configuration Register (SIU_PCR101)	16	R/W	See section	8.2.13/260
10C	Pad Configuration Register (SIU_PCR102)	16	R/W	See section	8.2.13/260
10E	Pad Configuration Register (SIU_PCR103)	16	R/W	See section	8.2.13/260
110	Pad Configuration Register (SIU_PCR104)	16	R/W	See section	8.2.13/260
112	Pad Configuration Register (SIU_PCR105)	16	R/W	See section	8.2.13/260
114	Pad Configuration Register (SIU_PCR106)	16	R/W	See section	8.2.13/260
116	Pad Configuration Register (SIU_PCR107)	16	R/W	See section	8.2.13/260
118	Pad Configuration Register (SIU_PCR108)	16	R/W	See section	8.2.13/260
11A	Pad Configuration Register (SIU_PCR109)	16	R/W	See section	8.2.13/260
11C	Pad Configuration Register (SIU_PCR110)	16	R/W	See section	8.2.13/260
11E	Pad Configuration Register (SIU_PCR111)	16	R/W	See section	8.2.13/260
120	Pad Configuration Register (SIU_PCR112)	16	R/W	See section	8.2.13/260
122	Pad Configuration Register (SIU_PCR113)	16	R/W	See section	8.2.13/260
124	Pad Configuration Register (SIU_PCR114)	16	R/W	See section	8.2.13/260
126	Pad Configuration Register (SIU_PCR115)	16	R/W	See section	8.2.13/260
128	Pad Configuration Register (SIU_PCR116)	16	R/W	See section	8.2.13/260
12A	Pad Configuration Register (SIU_PCR117)	16	R/W	See section	8.2.13/260
12C	Pad Configuration Register (SIU_PCR118)	16	R/W	See section	8.2.13/260
12E	Pad Configuration Register (SIU_PCR119)	16	R/W	See section	8.2.13/260
130	Pad Configuration Register (SIU_PCR120)	16	R/W	See section	8.2.13/260
132	Pad Configuration Register (SIU_PCR121)	16	R/W	See section	8.2.13/260
134	Pad Configuration Register (SIU_PCR122)	16	R/W	See section	8.2.13/260
136	Pad Configuration Register (SIU_PCR123)	16	R/W	See section	8.2.13/260
138	Pad Configuration Register (SIU_PCR124)	16	R/W	See section	8.2.13/260
13A	Pad Configuration Register (SIU_PCR125)	16	R/W	See section	8.2.13/260
13C	Pad Configuration Register (SIU_PCR126)	16	R/W	See section	8.2.13/260
13E	Pad Configuration Register (SIU_PCR127)	16	R/W	See section	8.2.13/260
140	Pad Configuration Register (SIU_PCR128)	16	R/W	See section	8.2.13/260
142	Pad Configuration Register (SIU_PCR129)	16	R/W	See section	8.2.13/260
144	Pad Configuration Register (SIU_PCR130)	16	R/W	See section	8.2.13/260
146	Pad Configuration Register (SIU_PCR131)	16	R/W	See section	8.2.13/260
148	Pad Configuration Register (SIU_PCR132)	16	R/W	See section	8.2.13/260
14A	Pad Configuration Register (SIU_PCR133)	16	R/W	See section	8.2.13/260
14C	Pad Configuration Register (SIU_PCR134)	16	R/W	See section	8.2.13/260
14E	Pad Configuration Register (SIU_PCR135)	16	R/W	See section	8.2.13/260

Table continues on the next page...

SIU memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
150	Pad Configuration Register (SIU_PCR136)	16	R/W	See section	8.2.13/260
152	Pad Configuration Register (SIU_PCR137)	16	R/W	See section	8.2.13/260
154	Pad Configuration Register (SIU_PCR138)	16	R/W	See section	8.2.13/260
156	Pad Configuration Register (SIU_PCR139)	16	R/W	See section	8.2.13/260
158	Pad Configuration Register (SIU_PCR140)	16	R/W	See section	8.2.13/260
15A	Pad Configuration Register (SIU_PCR141)	16	R/W	See section	8.2.13/260
15C	Pad Configuration Register (SIU_PCR142)	16	R/W	See section	8.2.13/260
15E	Pad Configuration Register (SIU_PCR143)	16	R/W	See section	8.2.13/260
160	Pad Configuration Register (SIU_PCR144)	16	R/W	See section	8.2.13/260
162	Pad Configuration Register (SIU_PCR145)	16	R/W	See section	8.2.13/260
164	Pad Configuration Register (SIU_PCR146)	16	R/W	See section	8.2.13/260
166	Pad Configuration Register (SIU_PCR147)	16	R/W	See section	8.2.13/260
168	Pad Configuration Register (SIU_PCR148)	16	R/W	See section	8.2.13/260
16A	Pad Configuration Register (SIU_PCR149)	16	R/W	See section	8.2.13/260
16C	Pad Configuration Register (SIU_PCR150)	16	R/W	See section	8.2.13/260
16E	Pad Configuration Register (SIU_PCR151)	16	R/W	See section	8.2.13/260
170	Pad Configuration Register (SIU_PCR152)	16	R/W	See section	8.2.13/260
172	Pad Configuration Register (SIU_PCR153)	16	R/W	See section	8.2.13/260
174	Pad Configuration Register (SIU_PCR154)	16	R/W	See section	8.2.13/260
176	Pad Configuration Register (SIU_PCR155)	16	R/W	See section	8.2.13/260
178	Pad Configuration Register (SIU_PCR156)	16	R/W	See section	8.2.13/260
17A	Pad Configuration Register (SIU_PCR157)	16	R/W	See section	8.2.13/260
17C	Pad Configuration Register (SIU_PCR158)	16	R/W	See section	8.2.13/260
17E	Pad Configuration Register (SIU_PCR159)	16	R/W	See section	8.2.13/260
180	Pad Configuration Register (SIU_PCR160)	16	R/W	See section	8.2.13/260
182	Pad Configuration Register (SIU_PCR161)	16	R/W	See section	8.2.13/260
184	Pad Configuration Register (SIU_PCR162)	16	R/W	See section	8.2.13/260
186	Pad Configuration Register (SIU_PCR163)	16	R/W	See section	8.2.13/260
188	Pad Configuration Register (SIU_PCR164)	16	R/W	See section	8.2.13/260
18A	Pad Configuration Register (SIU_PCR165)	16	R/W	See section	8.2.13/260
18C	Pad Configuration Register (SIU_PCR166)	16	R/W	See section	8.2.13/260
18E	Pad Configuration Register (SIU_PCR167)	16	R/W	See section	8.2.13/260
190	Pad Configuration Register (SIU_PCR168)	16	R/W	See section	8.2.13/260
192	Pad Configuration Register (SIU_PCR169)	16	R/W	See section	8.2.13/260
194	Pad Configuration Register (SIU_PCR170)	16	R/W	See section	8.2.13/260
196	Pad Configuration Register (SIU_PCR171)	16	R/W	See section	8.2.13/260
198	Pad Configuration Register (SIU_PCR172)	16	R/W	See section	8.2.13/260
19A	Pad Configuration Register (SIU_PCR173)	16	R/W	See section	8.2.13/260
19C	Pad Configuration Register (SIU_PCR174)	16	R/W	See section	8.2.13/260

Table continues on the next page...

SIU memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
19E	Pad Configuration Register (SIU_PCR175)	16	R/W	See section	8.2.13/260
1A0	Pad Configuration Register (SIU_PCR176)	16	R/W	See section	8.2.13/260
1A2	Pad Configuration Register (SIU_PCR177)	16	R/W	See section	8.2.13/260
1A4	Pad Configuration Register (SIU_PCR178)	16	R/W	See section	8.2.13/260
1A6	Pad Configuration Register (SIU_PCR179)	16	R/W	See section	8.2.13/260
1A8	Pad Configuration Register (SIU_PCR180)	16	R/W	See section	8.2.13/260
1AA	Pad Configuration Register (SIU_PCR181)	16	R/W	See section	8.2.13/260
1AC	Pad Configuration Register (SIU_PCR182)	16	R/W	See section	8.2.13/260
1AE	Pad Configuration Register (SIU_PCR183)	16	R/W	See section	8.2.13/260
1B0	Pad Configuration Register (SIU_PCR184)	16	R/W	See section	8.2.13/260
1B2	Pad Configuration Register (SIU_PCR185)	16	R/W	See section	8.2.13/260
1B4	Pad Configuration Register (SIU_PCR186)	16	R/W	See section	8.2.13/260
1B6	Pad Configuration Register (SIU_PCR187)	16	R/W	See section	8.2.13/260
1B8	Pad Configuration Register (SIU_PCR188)	16	R/W	See section	8.2.13/260
1BA	Pad Configuration Register (SIU_PCR189)	16	R/W	See section	8.2.13/260
1BC	Pad Configuration Register (SIU_PCR190)	16	R/W	See section	8.2.13/260
1BE	Pad Configuration Register (SIU_PCR191)	16	R/W	See section	8.2.13/260
1C0	Pad Configuration Register (SIU_PCR192)	16	R/W	See section	8.2.13/260
1C2	Pad Configuration Register (SIU_PCR193)	16	R/W	See section	8.2.13/260
1C4	Pad Configuration Register (SIU_PCR194)	16	R/W	See section	8.2.13/260
1C6	Pad Configuration Register (SIU_PCR195)	16	R/W	See section	8.2.13/260
1C8	Pad Configuration Register (SIU_PCR196)	16	R/W	See section	8.2.13/260
1CA	Pad Configuration Register (SIU_PCR197)	16	R/W	See section	8.2.13/260
1CC	Pad Configuration Register (SIU_PCR198)	16	R/W	See section	8.2.13/260
1CE	Pad Configuration Register (SIU_PCR199)	16	R/W	See section	8.2.13/260
1D0	Pad Configuration Register (SIU_PCR200)	16	R/W	See section	8.2.13/260
1D2	Pad Configuration Register (SIU_PCR201)	16	R/W	See section	8.2.13/260
1D4	Pad Configuration Register (SIU_PCR202)	16	R/W	See section	8.2.13/260
1D6	Pad Configuration Register (SIU_PCR203)	16	R/W	See section	8.2.13/260
1D8	Pad Configuration Register (SIU_PCR204)	16	R/W	See section	8.2.13/260
1DA	Pad Configuration Register (SIU_PCR205)	16	R/W	See section	8.2.13/260
1DC	Pad Configuration Register (SIU_PCR206)	16	R/W	See section	8.2.13/260
1DE	Pad Configuration Register (SIU_PCR207)	16	R/W	See section	8.2.13/260
1E0	Pad Configuration Register (SIU_PCR208)	16	R/W	See section	8.2.13/260
1E2	Pad Configuration Register (SIU_PCR209)	16	R/W	See section	8.2.13/260
1E4	Pad Configuration Register (SIU_PCR210)	16	R/W	See section	8.2.13/260
1E6	Pad Configuration Register (SIU_PCR211)	16	R/W	See section	8.2.13/260
1E8	Pad Configuration Register (SIU_PCR212)	16	R/W	See section	8.2.13/260
1EA	Pad Configuration Register (SIU_PCR213)	16	R/W	See section	8.2.13/260

Table continues on the next page...

SIU memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
1EC	Pad Configuration Register (SIU_PCR214)	16	R/W	See section	8.2.13/260
1EE	Pad Configuration Register (SIU_PCR215)	16	R/W	See section	8.2.13/260
1F0	Pad Configuration Register (SIU_PCR216)	16	R/W	See section	8.2.13/260
1F2	Pad Configuration Register (SIU_PCR217)	16	R/W	See section	8.2.13/260
1F4	Pad Configuration Register (SIU_PCR218)	16	R/W	See section	8.2.13/260
1F6	Pad Configuration Register (SIU_PCR219)	16	R/W	See section	8.2.13/260
1F8	Pad Configuration Register (SIU_PCR220)	16	R/W	See section	8.2.13/260
1FA	Pad Configuration Register (SIU_PCR221)	16	R/W	See section	8.2.13/260
1FC	Pad Configuration Register (SIU_PCR222)	16	R/W	See section	8.2.13/260
1FE	Pad Configuration Register (SIU_PCR223)	16	R/W	See section	8.2.13/260
200	Pad Configuration Register (SIU_PCR224)	16	R/W	See section	8.2.13/260
202	Pad Configuration Register (SIU_PCR225)	16	R/W	See section	8.2.13/260
204	Pad Configuration Register (SIU_PCR226)	16	R/W	See section	8.2.13/260
206	Pad Configuration Register (SIU_PCR227)	16	R/W	See section	8.2.13/260
208	Pad Configuration Register (SIU_PCR228)	16	R/W	See section	8.2.13/260
20A	Pad Configuration Register (SIU_PCR229)	16	R/W	See section	8.2.13/260
20C	Pad Configuration Register (SIU_PCR230)	16	R/W	See section	8.2.13/260
20E	Pad Configuration Register (SIU_PCR231)	16	R/W	See section	8.2.13/260
210	Pad Configuration Register (SIU_PCR232)	16	R/W	See section	8.2.13/260
212	Pad Configuration Register (SIU_PCR233)	16	R/W	See section	8.2.13/260
214	Pad Configuration Register (SIU_PCR234)	16	R/W	See section	8.2.13/260
216	Pad Configuration Register (SIU_PCR235)	16	R/W	See section	8.2.13/260
218	Pad Configuration Register (SIU_PCR236)	16	R/W	See section	8.2.13/260
21A	Pad Configuration Register (SIU_PCR237)	16	R/W	See section	8.2.13/260
21C	Pad Configuration Register (SIU_PCR238)	16	R/W	See section	8.2.13/260
21E	Pad Configuration Register (SIU_PCR239)	16	R/W	See section	8.2.13/260
220	Pad Configuration Register (SIU_PCR240)	16	R/W	See section	8.2.13/260
222	Pad Configuration Register (SIU_PCR241)	16	R/W	See section	8.2.13/260
224	Pad Configuration Register (SIU_PCR242)	16	R/W	See section	8.2.13/260
226	Pad Configuration Register (SIU_PCR243)	16	R/W	See section	8.2.13/260
228	Pad Configuration Register (SIU_PCR244)	16	R/W	See section	8.2.13/260
22A	Pad Configuration Register (SIU_PCR245)	16	R/W	See section	8.2.13/260
22C	Pad Configuration Register (SIU_PCR246)	16	R/W	See section	8.2.13/260
22E	Pad Configuration Register (SIU_PCR247)	16	R/W	See section	8.2.13/260
230	Pad Configuration Register (SIU_PCR248)	16	R/W	See section	8.2.13/260
232	Pad Configuration Register (SIU_PCR249)	16	R/W	See section	8.2.13/260
234	Pad Configuration Register (SIU_PCR250)	16	R/W	See section	8.2.13/260
236	Pad Configuration Register (SIU_PCR251)	16	R/W	See section	8.2.13/260
238	Pad Configuration Register (SIU_PCR252)	16	R/W	See section	8.2.13/260

Table continues on the next page...

SIU memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
23A	Pad Configuration Register (SIU_PCR253)	16	R/W	See section	8.2.13/260
23C	Pad Configuration Register (SIU_PCR254)	16	R/W	See section	8.2.13/260
23E	Pad Configuration Register (SIU_PCR255)	16	R/W	See section	8.2.13/260
240	Pad Configuration Register (SIU_PCR256)	16	R/W	See section	8.2.13/260
242	Pad Configuration Register (SIU_PCR257)	16	R/W	See section	8.2.13/260
244	Pad Configuration Register (SIU_PCR258)	16	R/W	See section	8.2.13/260
246	Pad Configuration Register (SIU_PCR259)	16	R/W	See section	8.2.13/260
248	Pad Configuration Register (SIU_PCR260)	16	R/W	See section	8.2.13/260
24A	Pad Configuration Register (SIU_PCR261)	16	R/W	See section	8.2.13/260
24C	Pad Configuration Register (SIU_PCR262)	16	R/W	See section	8.2.13/260
24E	Pad Configuration Register (SIU_PCR263)	16	R/W	See section	8.2.13/260
250	Pad Configuration Register (SIU_PCR264)	16	R/W	See section	8.2.13/260
252	Pad Configuration Register (SIU_PCR265)	16	R/W	See section	8.2.13/260
254	Pad Configuration Register (SIU_PCR266)	16	R/W	See section	8.2.13/260
256	Pad Configuration Register (SIU_PCR267)	16	R/W	See section	8.2.13/260
258	Pad Configuration Register (SIU_PCR268)	16	R/W	See section	8.2.13/260
25A	Pad Configuration Register (SIU_PCR269)	16	R/W	See section	8.2.13/260
25C	Pad Configuration Register (SIU_PCR270)	16	R/W	See section	8.2.13/260
25E	Pad Configuration Register (SIU_PCR271)	16	R/W	See section	8.2.13/260
260	Pad Configuration Register (SIU_PCR272)	16	R/W	See section	8.2.13/260
262	Pad Configuration Register (SIU_PCR273)	16	R/W	See section	8.2.13/260
264	Pad Configuration Register (SIU_PCR274)	16	R/W	See section	8.2.13/260
266	Pad Configuration Register (SIU_PCR275)	16	R/W	See section	8.2.13/260
268	Pad Configuration Register (SIU_PCR276)	16	R/W	See section	8.2.13/260
26A	Pad Configuration Register (SIU_PCR277)	16	R/W	See section	8.2.13/260
26C	Pad Configuration Register (SIU_PCR278)	16	R/W	See section	8.2.13/260
26E	Pad Configuration Register (SIU_PCR279)	16	R/W	See section	8.2.13/260
270	Pad Configuration Register (SIU_PCR280)	16	R/W	See section	8.2.13/260
272	Pad Configuration Register (SIU_PCR281)	16	R/W	See section	8.2.13/260
274	Pad Configuration Register (SIU_PCR282)	16	R/W	See section	8.2.13/260
276	Pad Configuration Register (SIU_PCR283)	16	R/W	See section	8.2.13/260
278	Pad Configuration Register (SIU_PCR284)	16	R/W	See section	8.2.13/260
27A	Pad Configuration Register (SIU_PCR285)	16	R/W	See section	8.2.13/260
27C	Pad Configuration Register (SIU_PCR286)	16	R/W	See section	8.2.13/260
27E	Pad Configuration Register (SIU_PCR287)	16	R/W	See section	8.2.13/260
280	Pad Configuration Register (SIU_PCR288)	16	R/W	See section	8.2.13/260
282	Pad Configuration Register (SIU_PCR289)	16	R/W	See section	8.2.13/260
284	Pad Configuration Register (SIU_PCR290)	16	R/W	See section	8.2.13/260
286	Pad Configuration Register (SIU_PCR291)	16	R/W	See section	8.2.13/260

Table continues on the next page...

SIU memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
288	Pad Configuration Register (SIU_PCR292)	16	R/W	See section	8.2.13/260
28A	Pad Configuration Register (SIU_PCR293)	16	R/W	See section	8.2.13/260
28C	Pad Configuration Register (SIU_PCR294)	16	R/W	See section	8.2.13/260
28E	Pad Configuration Register (SIU_PCR295)	16	R/W	See section	8.2.13/260
290	Pad Configuration Register (SIU_PCR296)	16	R/W	See section	8.2.13/260
292	Pad Configuration Register (SIU_PCR297)	16	R/W	See section	8.2.13/260
294	Pad Configuration Register (SIU_PCR298)	16	R/W	See section	8.2.13/260
296	Pad Configuration Register (SIU_PCR299)	16	R/W	See section	8.2.13/260
298	Pad Configuration Register (SIU_PCR300)	16	R/W	See section	8.2.13/260
29A	Pad Configuration Register (SIU_PCR301)	16	R/W	See section	8.2.13/260
29C	Pad Configuration Register (SIU_PCR302)	16	R/W	See section	8.2.13/260
29E	Pad Configuration Register (SIU_PCR303)	16	R/W	See section	8.2.13/260
2A0	Pad Configuration Register (SIU_PCR304)	16	R/W	See section	8.2.13/260
2A2	Pad Configuration Register (SIU_PCR305)	16	R/W	See section	8.2.13/260
2A4	Pad Configuration Register (SIU_PCR306)	16	R/W	See section	8.2.13/260
2A6	Pad Configuration Register (SIU_PCR307)	16	R/W	See section	8.2.13/260
2A8	Pad Configuration Register (SIU_PCR308)	16	R/W	See section	8.2.13/260
2AA	Pad Configuration Register (SIU_PCR309)	16	R/W	See section	8.2.13/260
2AC	Pad Configuration Register (SIU_PCR310)	16	R/W	See section	8.2.13/260
2AE	Pad Configuration Register (SIU_PCR311)	16	R/W	See section	8.2.13/260
2B0	Pad Configuration Register (SIU_PCR312)	16	R/W	See section	8.2.13/260
2B2	Pad Configuration Register (SIU_PCR313)	16	R/W	See section	8.2.13/260
2B4	Pad Configuration Register (SIU_PCR314)	16	R/W	See section	8.2.13/260
2B6	Pad Configuration Register (SIU_PCR315)	16	R/W	See section	8.2.13/260
2B8	Pad Configuration Register (SIU_PCR316)	16	R/W	See section	8.2.13/260
2BA	Pad Configuration Register (SIU_PCR317)	16	R/W	See section	8.2.13/260
2BC	Pad Configuration Register (SIU_PCR318)	16	R/W	See section	8.2.13/260
2BE	Pad Configuration Register (SIU_PCR319)	16	R/W	See section	8.2.13/260
2C0	Pad Configuration Register (SIU_PCR320)	16	R/W	See section	8.2.13/260
2C2	Pad Configuration Register (SIU_PCR321)	16	R/W	See section	8.2.13/260
2C4	Pad Configuration Register (SIU_PCR322)	16	R/W	See section	8.2.13/260
2C6	Pad Configuration Register (SIU_PCR323)	16	R/W	See section	8.2.13/260
2C8	Pad Configuration Register (SIU_PCR324)	16	R/W	See section	8.2.13/260
2CA	Pad Configuration Register (SIU_PCR325)	16	R/W	See section	8.2.13/260
2CC	Pad Configuration Register (SIU_PCR326)	16	R/W	See section	8.2.13/260
2CE	Pad Configuration Register (SIU_PCR327)	16	R/W	See section	8.2.13/260
2D0	Pad Configuration Register (SIU_PCR328)	16	R/W	See section	8.2.13/260
2D2	Pad Configuration Register (SIU_PCR329)	16	R/W	See section	8.2.13/260
2D4	Pad Configuration Register (SIU_PCR330)	16	R/W	See section	8.2.13/260

Table continues on the next page...

SIU memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
2D6	Pad Configuration Register (SIU_PCR331)	16	R/W	See section	8.2.13/260
2D8	Pad Configuration Register (SIU_PCR332)	16	R/W	See section	8.2.13/260
2DA	Pad Configuration Register (SIU_PCR333)	16	R/W	See section	8.2.13/260
2DC	Pad Configuration Register (SIU_PCR334)	16	R/W	See section	8.2.13/260
2DE	Pad Configuration Register (SIU_PCR335)	16	R/W	See section	8.2.13/260
2E0	Pad Configuration Register (SIU_PCR336)	16	R/W	See section	8.2.13/260
2E2	Pad Configuration Register (SIU_PCR337)	16	R/W	See section	8.2.13/260
2E4	Pad Configuration Register (SIU_PCR338)	16	R/W	See section	8.2.13/260
2E6	Pad Configuration Register (SIU_PCR339)	16	R/W	See section	8.2.13/260
2E8	Pad Configuration Register (SIU_PCR340)	16	R/W	See section	8.2.13/260
2EA	Pad Configuration Register (SIU_PCR341)	16	R/W	See section	8.2.13/260
2EC	Pad Configuration Register (SIU_PCR342)	16	R/W	See section	8.2.13/260
2EE	Pad Configuration Register (SIU_PCR343)	16	R/W	See section	8.2.13/260
2F0	Pad Configuration Register (SIU_PCR344)	16	R/W	See section	8.2.13/260
2F2	Pad Configuration Register (SIU_PCR345)	16	R/W	See section	8.2.13/260
2F4	Pad Configuration Register (SIU_PCR346)	16	R/W	See section	8.2.13/260
2F6	Pad Configuration Register (SIU_PCR347)	16	R/W	See section	8.2.13/260
2F8	Pad Configuration Register (SIU_PCR348)	16	R/W	See section	8.2.13/260
2FA	Pad Configuration Register (SIU_PCR349)	16	R/W	See section	8.2.13/260
2FC	Pad Configuration Register (SIU_PCR350)	16	R/W	See section	8.2.13/260
2FE	Pad Configuration Register (SIU_PCR351)	16	R/W	See section	8.2.13/260
300	Pad Configuration Register (SIU_PCR352)	16	R/W	See section	8.2.13/260
302	Pad Configuration Register (SIU_PCR353)	16	R/W	See section	8.2.13/260
304	Pad Configuration Register (SIU_PCR354)	16	R/W	See section	8.2.13/260
306	Pad Configuration Register (SIU_PCR355)	16	R/W	See section	8.2.13/260
308	Pad Configuration Register (SIU_PCR356)	16	R/W	See section	8.2.13/260
30A	Pad Configuration Register (SIU_PCR357)	16	R/W	See section	8.2.13/260
30C	Pad Configuration Register (SIU_PCR358)	16	R/W	See section	8.2.13/260
30E	Pad Configuration Register (SIU_PCR359)	16	R/W	See section	8.2.13/260
310	Pad Configuration Register (SIU_PCR360)	16	R/W	See section	8.2.13/260
312	Pad Configuration Register (SIU_PCR361)	16	R/W	See section	8.2.13/260
314	Pad Configuration Register (SIU_PCR362)	16	R/W	See section	8.2.13/260
316	Pad Configuration Register (SIU_PCR363)	16	R/W	See section	8.2.13/260
318	Pad Configuration Register (SIU_PCR364)	16	R/W	See section	8.2.13/260
31A	Pad Configuration Register (SIU_PCR365)	16	R/W	See section	8.2.13/260
31C	Pad Configuration Register (SIU_PCR366)	16	R/W	See section	8.2.13/260
31E	Pad Configuration Register (SIU_PCR367)	16	R/W	See section	8.2.13/260
320	Pad Configuration Register (SIU_PCR368)	16	R/W	See section	8.2.13/260
322	Pad Configuration Register (SIU_PCR369)	16	R/W	See section	8.2.13/260

Table continues on the next page...

SIU memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
324	Pad Configuration Register (SIU_PCR370)	16	R/W	See section	8.2.13/260
326	Pad Configuration Register (SIU_PCR371)	16	R/W	See section	8.2.13/260
328	Pad Configuration Register (SIU_PCR372)	16	R/W	See section	8.2.13/260
32A	Pad Configuration Register (SIU_PCR373)	16	R/W	See section	8.2.13/260
32C	Pad Configuration Register (SIU_PCR374)	16	R/W	See section	8.2.13/260
32E	Pad Configuration Register (SIU_PCR375)	16	R/W	See section	8.2.13/260
330	Pad Configuration Register (SIU_PCR376)	16	R/W	See section	8.2.13/260
332	Pad Configuration Register (SIU_PCR377)	16	R/W	See section	8.2.13/260
334	Pad Configuration Register (SIU_PCR378)	16	R/W	See section	8.2.13/260
336	Pad Configuration Register (SIU_PCR379)	16	R/W	See section	8.2.13/260
338	Pad Configuration Register (SIU_PCR380)	16	R/W	See section	8.2.13/260
33A	Pad Configuration Register (SIU_PCR381)	16	R/W	See section	8.2.13/260
33C	Pad Configuration Register (SIU_PCR382)	16	R/W	See section	8.2.13/260
33E	Pad Configuration Register (SIU_PCR383)	16	R/W	See section	8.2.13/260
340	Pad Configuration Register (SIU_PCR384)	16	R/W	See section	8.2.13/260
342	Pad Configuration Register (SIU_PCR385)	16	R/W	See section	8.2.13/260
344	Pad Configuration Register (SIU_PCR386)	16	R/W	See section	8.2.13/260
346	Pad Configuration Register (SIU_PCR387)	16	R/W	See section	8.2.13/260
348	Pad Configuration Register (SIU_PCR388)	16	R/W	See section	8.2.13/260
34A	Pad Configuration Register (SIU_PCR389)	16	R/W	See section	8.2.13/260
34C	Pad Configuration Register (SIU_PCR390)	16	R/W	See section	8.2.13/260
34E	Pad Configuration Register (SIU_PCR391)	16	R/W	See section	8.2.13/260
350	Pad Configuration Register (SIU_PCR392)	16	R/W	See section	8.2.13/260
352	Pad Configuration Register (SIU_PCR393)	16	R/W	See section	8.2.13/260
354	Pad Configuration Register (SIU_PCR394)	16	R/W	See section	8.2.13/260
356	Pad Configuration Register (SIU_PCR395)	16	R/W	See section	8.2.13/260
358	Pad Configuration Register (SIU_PCR396)	16	R/W	See section	8.2.13/260
35A	Pad Configuration Register (SIU_PCR397)	16	R/W	See section	8.2.13/260
35C	Pad Configuration Register (SIU_PCR398)	16	R/W	See section	8.2.13/260
35E	Pad Configuration Register (SIU_PCR399)	16	R/W	See section	8.2.13/260
360	Pad Configuration Register (SIU_PCR400)	16	R/W	See section	8.2.13/260
362	Pad Configuration Register (SIU_PCR401)	16	R/W	See section	8.2.13/260
364	Pad Configuration Register (SIU_PCR402)	16	R/W	See section	8.2.13/260
366	Pad Configuration Register (SIU_PCR403)	16	R/W	See section	8.2.13/260
368	Pad Configuration Register (SIU_PCR404)	16	R/W	See section	8.2.13/260
36A	Pad Configuration Register (SIU_PCR405)	16	R/W	See section	8.2.13/260
36C	Pad Configuration Register (SIU_PCR406)	16	R/W	See section	8.2.13/260
36E	Pad Configuration Register (SIU_PCR407)	16	R/W	See section	8.2.13/260
370	Pad Configuration Register (SIU_PCR408)	16	R/W	See section	8.2.13/260

Table continues on the next page...

SIU memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
372	Pad Configuration Register (SIU_PCR409)	16	R/W	See section	8.2.13/260
374	Pad Configuration Register (SIU_PCR410)	16	R/W	See section	8.2.13/260
376	Pad Configuration Register (SIU_PCR411)	16	R/W	See section	8.2.13/260
378	Pad Configuration Register (SIU_PCR412)	16	R/W	See section	8.2.13/260
37A	Pad Configuration Register (SIU_PCR413)	16	R/W	See section	8.2.13/260
37C	Pad Configuration Register (SIU_PCR414)	16	R/W	See section	8.2.13/260
37E	Pad Configuration Register (SIU_PCR415)	16	R/W	See section	8.2.13/260
380	Pad Configuration Register (SIU_PCR416)	16	R/W	See section	8.2.13/260
382	Pad Configuration Register (SIU_PCR417)	16	R/W	See section	8.2.13/260
384	Pad Configuration Register (SIU_PCR418)	16	R/W	See section	8.2.13/260
386	Pad Configuration Register (SIU_PCR419)	16	R/W	See section	8.2.13/260
388	Pad Configuration Register (SIU_PCR420)	16	R/W	See section	8.2.13/260
38A	Pad Configuration Register (SIU_PCR421)	16	R/W	See section	8.2.13/260
38C	Pad Configuration Register (SIU_PCR422)	16	R/W	See section	8.2.13/260
38E	Pad Configuration Register (SIU_PCR423)	16	R/W	See section	8.2.13/260
390	Pad Configuration Register (SIU_PCR424)	16	R/W	See section	8.2.13/260
392	Pad Configuration Register (SIU_PCR425)	16	R/W	See section	8.2.13/260
394	Pad Configuration Register (SIU_PCR426)	16	R/W	See section	8.2.13/260
396	Pad Configuration Register (SIU_PCR427)	16	R/W	See section	8.2.13/260
398	Pad Configuration Register (SIU_PCR428)	16	R/W	See section	8.2.13/260
39A	Pad Configuration Register (SIU_PCR429)	16	R/W	See section	8.2.13/260
39C	Pad Configuration Register (SIU_PCR430)	16	R/W	See section	8.2.13/260
39E	Pad Configuration Register (SIU_PCR431)	16	R/W	See section	8.2.13/260
3A0	Pad Configuration Register (SIU_PCR432)	16	R/W	See section	8.2.13/260
3A2	Pad Configuration Register (SIU_PCR433)	16	R/W	See section	8.2.13/260
3A4	Pad Configuration Register (SIU_PCR434)	16	R/W	See section	8.2.13/260
3A6	Pad Configuration Register (SIU_PCR435)	16	R/W	See section	8.2.13/260
3A8	Pad Configuration Register (SIU_PCR436)	16	R/W	See section	8.2.13/260
3AA	Pad Configuration Register (SIU_PCR437)	16	R/W	See section	8.2.13/260
3AC	Pad Configuration Register (SIU_PCR438)	16	R/W	See section	8.2.13/260
3AE	Pad Configuration Register (SIU_PCR439)	16	R/W	See section	8.2.13/260
3B0	Pad Configuration Register (SIU_PCR440)	16	R/W	See section	8.2.13/260
3B2	Pad Configuration Register (SIU_PCR441)	16	R/W	See section	8.2.13/260
3B4	Pad Configuration Register (SIU_PCR442)	16	R/W	See section	8.2.13/260
3B6	Pad Configuration Register (SIU_PCR443)	16	R/W	See section	8.2.13/260
3B8	Pad Configuration Register (SIU_PCR444)	16	R/W	See section	8.2.13/260
3BA	Pad Configuration Register (SIU_PCR445)	16	R/W	See section	8.2.13/260
3BC	Pad Configuration Register (SIU_PCR446)	16	R/W	See section	8.2.13/260
3BE	Pad Configuration Register (SIU_PCR447)	16	R/W	See section	8.2.13/260

Table continues on the next page...

SIU memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
3C0	Pad Configuration Register (SIU_PCR448)	16	R/W	See section	8.2.13/260
3C2	Pad Configuration Register (SIU_PCR449)	16	R/W	See section	8.2.13/260
3C4	Pad Configuration Register (SIU_PCR450)	16	R/W	See section	8.2.13/260
3C6	Pad Configuration Register (SIU_PCR451)	16	R/W	See section	8.2.13/260
3C8	Pad Configuration Register (SIU_PCR452)	16	R/W	See section	8.2.13/260
3CA	Pad Configuration Register (SIU_PCR453)	16	R/W	See section	8.2.13/260
3CC	Pad Configuration Register (SIU_PCR454)	16	R/W	See section	8.2.13/260
3CE	Pad Configuration Register (SIU_PCR455)	16	R/W	See section	8.2.13/260
3D0	Pad Configuration Register (SIU_PCR456)	16	R/W	See section	8.2.13/260
3D2	Pad Configuration Register (SIU_PCR457)	16	R/W	See section	8.2.13/260
3D4	Pad Configuration Register (SIU_PCR458)	16	R/W	See section	8.2.13/260
3D6	Pad Configuration Register (SIU_PCR459)	16	R/W	See section	8.2.13/260
3D8	Pad Configuration Register (SIU_PCR460)	16	R/W	See section	8.2.13/260
3DA	Pad Configuration Register (SIU_PCR461)	16	R/W	See section	8.2.13/260
3DC	Pad Configuration Register (SIU_PCR462)	16	R/W	See section	8.2.13/260
3DE	Pad Configuration Register (SIU_PCR463)	16	R/W	See section	8.2.13/260
3E0	Pad Configuration Register (SIU_PCR464)	16	R/W	See section	8.2.13/260
3E2	Pad Configuration Register (SIU_PCR465)	16	R/W	See section	8.2.13/260
3E4	Pad Configuration Register (SIU_PCR466)	16	R/W	See section	8.2.13/260
3E6	Pad Configuration Register (SIU_PCR467)	16	R/W	See section	8.2.13/260
3E8	Pad Configuration Register (SIU_PCR468)	16	R/W	See section	8.2.13/260
3EA	Pad Configuration Register (SIU_PCR469)	16	R/W	See section	8.2.13/260
3EC	Pad Configuration Register (SIU_PCR470)	16	R/W	See section	8.2.13/260
3EE	Pad Configuration Register (SIU_PCR471)	16	R/W	See section	8.2.13/260
3F0	Pad Configuration Register (SIU_PCR472)	16	R/W	See section	8.2.13/260
3F2	Pad Configuration Register (SIU_PCR473)	16	R/W	See section	8.2.13/260
3F4	Pad Configuration Register (SIU_PCR474)	16	R/W	See section	8.2.13/260
3F6	Pad Configuration Register (SIU_PCR475)	16	R/W	See section	8.2.13/260
3F8	Pad Configuration Register (SIU_PCR476)	16	R/W	See section	8.2.13/260
3FA	Pad Configuration Register (SIU_PCR477)	16	R/W	See section	8.2.13/260
3FC	Pad Configuration Register (SIU_PCR478)	16	R/W	See section	8.2.13/260
3FE	Pad Configuration Register (SIU_PCR479)	16	R/W	See section	8.2.13/260
400	Pad Configuration Register (SIU_PCR480)	16	R/W	See section	8.2.13/260
402	Pad Configuration Register (SIU_PCR481)	16	R/W	See section	8.2.13/260
404	Pad Configuration Register (SIU_PCR482)	16	R/W	See section	8.2.13/260
406	Pad Configuration Register (SIU_PCR483)	16	R/W	See section	8.2.13/260
408	Pad Configuration Register (SIU_PCR484)	16	R/W	See section	8.2.13/260
40A	Pad Configuration Register (SIU_PCR485)	16	R/W	See section	8.2.13/260
40C	Pad Configuration Register (SIU_PCR486)	16	R/W	See section	8.2.13/260

Table continues on the next page...

SIU memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
40E	Pad Configuration Register (SIU_PCR487)	16	R/W	See section	8.2.13/260
410	Pad Configuration Register (SIU_PCR488)	16	R/W	See section	8.2.13/260
412	Pad Configuration Register (SIU_PCR489)	16	R/W	See section	8.2.13/260
414	Pad Configuration Register (SIU_PCR490)	16	R/W	See section	8.2.13/260
416	Pad Configuration Register (SIU_PCR491)	16	R/W	See section	8.2.13/260
64B	GPIO Pin Data Output Register (SIU_GPDO75)	8	R/W	00h	8.2.14/263
64C	GPIO Pin Data Output Register (SIU_GPDO76)	8	R/W	00h	8.2.14/263
64D	GPIO Pin Data Output Register (SIU_GPDO77)	8	R/W	00h	8.2.14/263
64E	GPIO Pin Data Output Register (SIU_GPDO78)	8	R/W	00h	8.2.14/263
64F	GPIO Pin Data Output Register (SIU_GPDO79)	8	R/W	00h	8.2.14/263
650	GPIO Pin Data Output Register (SIU_GPDO80)	8	R/W	00h	8.2.14/263
651	GPIO Pin Data Output Register (SIU_GPDO81)	8	R/W	00h	8.2.14/263
652	GPIO Pin Data Output Register (SIU_GPDO82)	8	R/W	00h	8.2.14/263
653	GPIO Pin Data Output Register (SIU_GPDO83)	8	R/W	00h	8.2.14/263
654	GPIO Pin Data Output Register (SIU_GPDO84)	8	R/W	00h	8.2.14/263
655	GPIO Pin Data Output Register (SIU_GPDO85)	8	R/W	00h	8.2.14/263
656	GPIO Pin Data Output Register (SIU_GPDO86)	8	R/W	00h	8.2.14/263
657	GPIO Pin Data Output Register (SIU_GPDO87)	8	R/W	00h	8.2.14/263
658	GPIO Pin Data Output Register (SIU_GPDO88)	8	R/W	00h	8.2.14/263
659	GPIO Pin Data Output Register (SIU_GPDO89)	8	R/W	00h	8.2.14/263
65A	GPIO Pin Data Output Register (SIU_GPDO90)	8	R/W	00h	8.2.14/263
65B	GPIO Pin Data Output Register (SIU_GPDO91)	8	R/W	00h	8.2.14/263
65C	GPIO Pin Data Output Register (SIU_GPDO92)	8	R/W	00h	8.2.14/263
65D	GPIO Pin Data Output Register (SIU_GPDO93)	8	R/W	00h	8.2.14/263
65E	GPIO Pin Data Output Register (SIU_GPDO94)	8	R/W	00h	8.2.14/263
65F	GPIO Pin Data Output Register (SIU_GPDO95)	8	R/W	00h	8.2.14/263
660	GPIO Pin Data Output Register (SIU_GPDO96)	8	R/W	00h	8.2.14/263
661	GPIO Pin Data Output Register (SIU_GPDO97)	8	R/W	00h	8.2.14/263
662	GPIO Pin Data Output Register (SIU_GPDO98)	8	R/W	00h	8.2.14/263
663	GPIO Pin Data Output Register (SIU_GPDO99)	8	R/W	00h	8.2.14/263
664	GPIO Pin Data Output Register (SIU_GPDO100)	8	R/W	00h	8.2.14/263
665	GPIO Pin Data Output Register (SIU_GPDO101)	8	R/W	00h	8.2.14/263
666	GPIO Pin Data Output Register (SIU_GPDO102)	8	R/W	00h	8.2.14/263
667	GPIO Pin Data Output Register (SIU_GPDO103)	8	R/W	00h	8.2.14/263
668	GPIO Pin Data Output Register (SIU_GPDO104)	8	R/W	00h	8.2.14/263
669	GPIO Pin Data Output Register (SIU_GPDO105)	8	R/W	00h	8.2.14/263
66A	GPIO Pin Data Output Register (SIU_GPDO106)	8	R/W	00h	8.2.14/263
66B	GPIO Pin Data Output Register (SIU_GPDO107)	8	R/W	00h	8.2.14/263
66C	GPIO Pin Data Output Register (SIU_GPDO108)	8	R/W	00h	8.2.14/263

Table continues on the next page...

SIU memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
66D	GPIO Pin Data Output Register (SIU_GPDO109)	8	R/W	00h	8.2.14/263
66E	GPIO Pin Data Output Register (SIU_GPDO110)	8	R/W	00h	8.2.14/263
66F	GPIO Pin Data Output Register (SIU_GPDO111)	8	R/W	00h	8.2.14/263
670	GPIO Pin Data Output Register (SIU_GPDO112)	8	R/W	00h	8.2.14/263
671	GPIO Pin Data Output Register (SIU_GPDO113)	8	R/W	00h	8.2.14/263
672	GPIO Pin Data Output Register (SIU_GPDO114)	8	R/W	00h	8.2.14/263
673	GPIO Pin Data Output Register (SIU_GPDO115)	8	R/W	00h	8.2.14/263
674	GPIO Pin Data Output Register (SIU_GPDO116)	8	R/W	00h	8.2.14/263
675	GPIO Pin Data Output Register (SIU_GPDO117)	8	R/W	00h	8.2.14/263
676	GPIO Pin Data Output Register (SIU_GPDO118)	8	R/W	00h	8.2.14/263
677	GPIO Pin Data Output Register (SIU_GPDO119)	8	R/W	00h	8.2.14/263
678	GPIO Pin Data Output Register (SIU_GPDO120)	8	R/W	00h	8.2.14/263
679	GPIO Pin Data Output Register (SIU_GPDO121)	8	R/W	00h	8.2.14/263
67A	GPIO Pin Data Output Register (SIU_GPDO122)	8	R/W	00h	8.2.14/263
67B	GPIO Pin Data Output Register (SIU_GPDO123)	8	R/W	00h	8.2.14/263
67C	GPIO Pin Data Output Register (SIU_GPDO124)	8	R/W	00h	8.2.14/263
67D	GPIO Pin Data Output Register (SIU_GPDO125)	8	R/W	00h	8.2.14/263
67E	GPIO Pin Data Output Register (SIU_GPDO126)	8	R/W	00h	8.2.14/263
67F	GPIO Pin Data Output Register (SIU_GPDO127)	8	R/W	00h	8.2.14/263
680	GPIO Pin Data Output Register (SIU_GPDO128)	8	R/W	00h	8.2.14/263
681	GPIO Pin Data Output Register (SIU_GPDO129)	8	R/W	00h	8.2.14/263
682	GPIO Pin Data Output Register (SIU_GPDO130)	8	R/W	00h	8.2.14/263
683	GPIO Pin Data Output Register (SIU_GPDO131)	8	R/W	00h	8.2.14/263
684	GPIO Pin Data Output Register (SIU_GPDO132)	8	R/W	00h	8.2.14/263
685	GPIO Pin Data Output Register (SIU_GPDO133)	8	R/W	00h	8.2.14/263
686	GPIO Pin Data Output Register (SIU_GPDO134)	8	R/W	00h	8.2.14/263
687	GPIO Pin Data Output Register (SIU_GPDO135)	8	R/W	00h	8.2.14/263
688	GPIO Pin Data Output Register (SIU_GPDO136)	8	R/W	00h	8.2.14/263
689	GPIO Pin Data Output Register (SIU_GPDO137)	8	R/W	00h	8.2.14/263
68A	GPIO Pin Data Output Register (SIU_GPDO138)	8	R/W	00h	8.2.14/263
68B	GPIO Pin Data Output Register (SIU_GPDO139)	8	R/W	00h	8.2.14/263
68C	GPIO Pin Data Output Register (SIU_GPDO140)	8	R/W	00h	8.2.14/263
68D	GPIO Pin Data Output Register (SIU_GPDO141)	8	R/W	00h	8.2.14/263
68E	GPIO Pin Data Output Register (SIU_GPDO142)	8	R/W	00h	8.2.14/263
68F	GPIO Pin Data Output Register (SIU_GPDO143)	8	R/W	00h	8.2.14/263
690	GPIO Pin Data Output Register (SIU_GPDO144)	8	R/W	00h	8.2.14/263
691	GPIO Pin Data Output Register (SIU_GPDO145)	8	R/W	00h	8.2.14/263
692	GPIO Pin Data Output Register (SIU_GPDO146)	8	R/W	00h	8.2.14/263
693	GPIO Pin Data Output Register (SIU_GPDO147)	8	R/W	00h	8.2.14/263

Table continues on the next page...

SIU memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
694	GPIO Pin Data Output Register (SIU_GPDO148)	8	R/W	00h	8.2.14/263
695	GPIO Pin Data Output Register (SIU_GPDO149)	8	R/W	00h	8.2.14/263
696	GPIO Pin Data Output Register (SIU_GPDO150)	8	R/W	00h	8.2.14/263
697	GPIO Pin Data Output Register (SIU_GPDO151)	8	R/W	00h	8.2.14/263
698	GPIO Pin Data Output Register (SIU_GPDO152)	8	R/W	00h	8.2.14/263
699	GPIO Pin Data Output Register (SIU_GPDO153)	8	R/W	00h	8.2.14/263
69A	GPIO Pin Data Output Register (SIU_GPDO154)	8	R/W	00h	8.2.14/263
69B	GPIO Pin Data Output Register (SIU_GPDO155)	8	R/W	00h	8.2.14/263
69C	GPIO Pin Data Output Register (SIU_GPDO156)	8	R/W	00h	8.2.14/263
69D	GPIO Pin Data Output Register (SIU_GPDO157)	8	R/W	00h	8.2.14/263
69E	GPIO Pin Data Output Register (SIU_GPDO158)	8	R/W	00h	8.2.14/263
69F	GPIO Pin Data Output Register (SIU_GPDO159)	8	R/W	00h	8.2.14/263
6A0	GPIO Pin Data Output Register (SIU_GPDO160)	8	R/W	00h	8.2.14/263
6A1	GPIO Pin Data Output Register (SIU_GPDO161)	8	R/W	00h	8.2.14/263
6A2	GPIO Pin Data Output Register (SIU_GPDO162)	8	R/W	00h	8.2.14/263
6A3	GPIO Pin Data Output Register (SIU_GPDO163)	8	R/W	00h	8.2.14/263
6A4	GPIO Pin Data Output Register (SIU_GPDO164)	8	R/W	00h	8.2.14/263
6A5	GPIO Pin Data Output Register (SIU_GPDO165)	8	R/W	00h	8.2.14/263
6A6	GPIO Pin Data Output Register (SIU_GPDO166)	8	R/W	00h	8.2.14/263
6A7	GPIO Pin Data Output Register (SIU_GPDO167)	8	R/W	00h	8.2.14/263
6A8	GPIO Pin Data Output Register (SIU_GPDO168)	8	R/W	00h	8.2.14/263
6A9	GPIO Pin Data Output Register (SIU_GPDO169)	8	R/W	00h	8.2.14/263
6AA	GPIO Pin Data Output Register (SIU_GPDO170)	8	R/W	00h	8.2.14/263
6AB	GPIO Pin Data Output Register (SIU_GPDO171)	8	R/W	00h	8.2.14/263
6AC	GPIO Pin Data Output Register (SIU_GPDO172)	8	R/W	00h	8.2.14/263
6AD	GPIO Pin Data Output Register (SIU_GPDO173)	8	R/W	00h	8.2.14/263
6AE	GPIO Pin Data Output Register (SIU_GPDO174)	8	R/W	00h	8.2.14/263
6AF	GPIO Pin Data Output Register (SIU_GPDO175)	8	R/W	00h	8.2.14/263
6B0	GPIO Pin Data Output Register (SIU_GPDO176)	8	R/W	00h	8.2.14/263
6B1	GPIO Pin Data Output Register (SIU_GPDO177)	8	R/W	00h	8.2.14/263
6B2	GPIO Pin Data Output Register (SIU_GPDO178)	8	R/W	00h	8.2.14/263
6B3	GPIO Pin Data Output Register (SIU_GPDO179)	8	R/W	00h	8.2.14/263
6B4	GPIO Pin Data Output Register (SIU_GPDO180)	8	R/W	00h	8.2.14/263
6B5	GPIO Pin Data Output Register (SIU_GPDO181)	8	R/W	00h	8.2.14/263
6B6	GPIO Pin Data Output Register (SIU_GPDO182)	8	R/W	00h	8.2.14/263
6B7	GPIO Pin Data Output Register (SIU_GPDO183)	8	R/W	00h	8.2.14/263
6B8	GPIO Pin Data Output Register (SIU_GPDO184)	8	R/W	00h	8.2.14/263
6B9	GPIO Pin Data Output Register (SIU_GPDO185)	8	R/W	00h	8.2.14/263
6BA	GPIO Pin Data Output Register (SIU_GPDO186)	8	R/W	00h	8.2.14/263

Table continues on the next page...

SIU memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
6BB	GPIO Pin Data Output Register (SIU_GPDO187)	8	R/W	00h	8.2.14/263
6BC	GPIO Pin Data Output Register (SIU_GPDO188)	8	R/W	00h	8.2.14/263
6BD	GPIO Pin Data Output Register (SIU_GPDO189)	8	R/W	00h	8.2.14/263
6BE	GPIO Pin Data Output Register (SIU_GPDO190)	8	R/W	00h	8.2.14/263
6BF	GPIO Pin Data Output Register (SIU_GPDO191)	8	R/W	00h	8.2.14/263
6C0	GPIO Pin Data Output Register (SIU_GPDO192)	8	R/W	00h	8.2.14/263
6C1	GPIO Pin Data Output Register (SIU_GPDO193)	8	R/W	00h	8.2.14/263
6C2	GPIO Pin Data Output Register (SIU_GPDO194)	8	R/W	00h	8.2.14/263
6C3	GPIO Pin Data Output Register (SIU_GPDO195)	8	R/W	00h	8.2.14/263
6C4	GPIO Pin Data Output Register (SIU_GPDO196)	8	R/W	00h	8.2.14/263
6C5	GPIO Pin Data Output Register (SIU_GPDO197)	8	R/W	00h	8.2.14/263
6C6	GPIO Pin Data Output Register (SIU_GPDO198)	8	R/W	00h	8.2.14/263
6C7	GPIO Pin Data Output Register (SIU_GPDO199)	8	R/W	00h	8.2.14/263
6C8	GPIO Pin Data Output Register (SIU_GPDO200)	8	R/W	00h	8.2.14/263
6C9	GPIO Pin Data Output Register (SIU_GPDO201)	8	R/W	00h	8.2.14/263
6CA	GPIO Pin Data Output Register (SIU_GPDO202)	8	R/W	00h	8.2.14/263
6CB	GPIO Pin Data Output Register (SIU_GPDO203)	8	R/W	00h	8.2.14/263
6CC	GPIO Pin Data Output Register (SIU_GPDO204)	8	R/W	00h	8.2.14/263
6CD	GPIO Pin Data Output Register (SIU_GPDO205)	8	R/W	00h	8.2.14/263
6CE	GPIO Pin Data Output Register (SIU_GPDO206)	8	R/W	00h	8.2.14/263
6CF	GPIO Pin Data Output Register (SIU_GPDO207)	8	R/W	00h	8.2.14/263
6D0	GPIO Pin Data Output Register (SIU_GPDO208)	8	R/W	00h	8.2.14/263
6D1	GPIO Pin Data Output Register (SIU_GPDO209)	8	R/W	00h	8.2.14/263
6D2	GPIO Pin Data Output Register (SIU_GPDO210)	8	R/W	00h	8.2.14/263
6D3	GPIO Pin Data Output Register (SIU_GPDO211)	8	R/W	00h	8.2.14/263
6D4	GPIO Pin Data Output Register (SIU_GPDO212)	8	R/W	00h	8.2.14/263
6D5	GPIO Pin Data Output Register (SIU_GPDO213)	8	R/W	00h	8.2.14/263
6D6	GPIO Pin Data Output Register (SIU_GPDO214)	8	R/W	00h	8.2.14/263
6D7	GPIO Pin Data Output Register (SIU_GPDO215)	8	R/W	00h	8.2.14/263
6D8	GPIO Pin Data Output Register (SIU_GPDO216)	8	R/W	00h	8.2.14/263
6D9	GPIO Pin Data Output Register (SIU_GPDO217)	8	R/W	00h	8.2.14/263
6DA	GPIO Pin Data Output Register (SIU_GPDO218)	8	R/W	00h	8.2.14/263
6DB	GPIO Pin Data Output Register (SIU_GPDO219)	8	R/W	00h	8.2.14/263
6DC	GPIO Pin Data Output Register (SIU_GPDO220)	8	R/W	00h	8.2.14/263
6DD	GPIO Pin Data Output Register (SIU_GPDO221)	8	R/W	00h	8.2.14/263
6DE	GPIO Pin Data Output Register (SIU_GPDO222)	8	R/W	00h	8.2.14/263
6DF	GPIO Pin Data Output Register (SIU_GPDO223)	8	R/W	00h	8.2.14/263
6E0	GPIO Pin Data Output Register (SIU_GPDO224)	8	R/W	00h	8.2.14/263
6E1	GPIO Pin Data Output Register (SIU_GPDO225)	8	R/W	00h	8.2.14/263

Table continues on the next page...

SIU memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
6E2	GPIO Pin Data Output Register (SIU_GPDO226)	8	R/W	00h	8.2.14/263
6E3	GPIO Pin Data Output Register (SIU_GPDO227)	8	R/W	00h	8.2.14/263
6E4	GPIO Pin Data Output Register (SIU_GPDO228)	8	R/W	00h	8.2.14/263
6E5	GPIO Pin Data Output Register (SIU_GPDO229)	8	R/W	00h	8.2.14/263
6E6	GPIO Pin Data Output Register (SIU_GPDO230)	8	R/W	00h	8.2.14/263
6E7	GPIO Pin Data Output Register (SIU_GPDO231)	8	R/W	00h	8.2.14/263
6E8	GPIO Pin Data Output Register (SIU_GPDO232)	8	R/W	00h	8.2.14/263
6E9	GPIO Pin Data Output Register (SIU_GPDO233)	8	R/W	00h	8.2.14/263
6EA	GPIO Pin Data Output Register (SIU_GPDO234)	8	R/W	00h	8.2.14/263
6EB	GPIO Pin Data Output Register (SIU_GPDO235)	8	R/W	00h	8.2.14/263
6EC	GPIO Pin Data Output Register (SIU_GPDO236)	8	R/W	00h	8.2.14/263
6ED	GPIO Pin Data Output Register (SIU_GPDO237)	8	R/W	00h	8.2.14/263
6EE	GPIO Pin Data Output Register (SIU_GPDO238)	8	R/W	00h	8.2.14/263
6EF	GPIO Pin Data Output Register (SIU_GPDO239)	8	R/W	00h	8.2.14/263
6F0	GPIO Pin Data Output Register (SIU_GPDO240)	8	R/W	00h	8.2.14/263
6F1	GPIO Pin Data Output Register (SIU_GPDO241)	8	R/W	00h	8.2.14/263
6F2	GPIO Pin Data Output Register (SIU_GPDO242)	8	R/W	00h	8.2.14/263
6F3	GPIO Pin Data Output Register (SIU_GPDO243)	8	R/W	00h	8.2.14/263
6F4	GPIO Pin Data Output Register (SIU_GPDO244)	8	R/W	00h	8.2.14/263
6F5	GPIO Pin Data Output Register (SIU_GPDO245)	8	R/W	00h	8.2.14/263
6F6	GPIO Pin Data Output Register (SIU_GPDO246)	8	R/W	00h	8.2.14/263
6F7	GPIO Pin Data Output Register (SIU_GPDO247)	8	R/W	00h	8.2.14/263
6F8	GPIO Pin Data Output Register (SIU_GPDO248)	8	R/W	00h	8.2.14/263
6F9	GPIO Pin Data Output Register (SIU_GPDO249)	8	R/W	00h	8.2.14/263
6FA	GPIO Pin Data Output Register (SIU_GPDO250)	8	R/W	00h	8.2.14/263
6FB	GPIO Pin Data Output Register (SIU_GPDO251)	8	R/W	00h	8.2.14/263
6FC	GPIO Pin Data Output Register (SIU_GPDO252)	8	R/W	00h	8.2.14/263
6FD	GPIO Pin Data Output Register (SIU_GPDO253)	8	R/W	00h	8.2.14/263
6FE	GPIO Pin Data Output Register (SIU_GPDO254)	8	R/W	00h	8.2.14/263
6FF	GPIO Pin Data Output Register (SIU_GPDO255)	8	R/W	00h	8.2.14/263
700	GPIO Pin Data Output Register (SIU_GPDO256)	8	R/W	00h	8.2.14/263
701	GPIO Pin Data Output Register (SIU_GPDO257)	8	R/W	00h	8.2.14/263
702	GPIO Pin Data Output Register (SIU_GPDO258)	8	R/W	00h	8.2.14/263
703	GPIO Pin Data Output Register (SIU_GPDO259)	8	R/W	00h	8.2.14/263
704	GPIO Pin Data Output Register (SIU_GPDO260)	8	R/W	00h	8.2.14/263
705	GPIO Pin Data Output Register (SIU_GPDO261)	8	R/W	00h	8.2.14/263
706	GPIO Pin Data Output Register (SIU_GPDO262)	8	R/W	00h	8.2.14/263
707	GPIO Pin Data Output Register (SIU_GPDO263)	8	R/W	00h	8.2.14/263
708	GPIO Pin Data Output Register (SIU_GPDO264)	8	R/W	00h	8.2.14/263

Table continues on the next page...

SIU memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
709	GPIO Pin Data Output Register (SIU_GPDO265)	8	R/W	00h	8.2.14/263
70A	GPIO Pin Data Output Register (SIU_GPDO266)	8	R/W	00h	8.2.14/263
70B	GPIO Pin Data Output Register (SIU_GPDO267)	8	R/W	00h	8.2.14/263
70C	GPIO Pin Data Output Register (SIU_GPDO268)	8	R/W	00h	8.2.14/263
70D	GPIO Pin Data Output Register (SIU_GPDO269)	8	R/W	00h	8.2.14/263
70E	GPIO Pin Data Output Register (SIU_GPDO270)	8	R/W	00h	8.2.14/263
70F	GPIO Pin Data Output Register (SIU_GPDO271)	8	R/W	00h	8.2.14/263
710	GPIO Pin Data Output Register (SIU_GPDO272)	8	R/W	00h	8.2.14/263
711	GPIO Pin Data Output Register (SIU_GPDO273)	8	R/W	00h	8.2.14/263
712	GPIO Pin Data Output Register (SIU_GPDO274)	8	R/W	00h	8.2.14/263
713	GPIO Pin Data Output Register (SIU_GPDO275)	8	R/W	00h	8.2.14/263
714	GPIO Pin Data Output Register (SIU_GPDO276)	8	R/W	00h	8.2.14/263
715	GPIO Pin Data Output Register (SIU_GPDO277)	8	R/W	00h	8.2.14/263
716	GPIO Pin Data Output Register (SIU_GPDO278)	8	R/W	00h	8.2.14/263
717	GPIO Pin Data Output Register (SIU_GPDO279)	8	R/W	00h	8.2.14/263
718	GPIO Pin Data Output Register (SIU_GPDO280)	8	R/W	00h	8.2.14/263
719	GPIO Pin Data Output Register (SIU_GPDO281)	8	R/W	00h	8.2.14/263
71A	GPIO Pin Data Output Register (SIU_GPDO282)	8	R/W	00h	8.2.14/263
71B	GPIO Pin Data Output Register (SIU_GPDO283)	8	R/W	00h	8.2.14/263
71C	GPIO Pin Data Output Register (SIU_GPDO284)	8	R/W	00h	8.2.14/263
71D	GPIO Pin Data Output Register (SIU_GPDO285)	8	R/W	00h	8.2.14/263
71E	GPIO Pin Data Output Register (SIU_GPDO286)	8	R/W	00h	8.2.14/263
71F	GPIO Pin Data Output Register (SIU_GPDO287)	8	R/W	00h	8.2.14/263
720	GPIO Pin Data Output Register (SIU_GPDO288)	8	R/W	00h	8.2.14/263
721	GPIO Pin Data Output Register (SIU_GPDO289)	8	R/W	00h	8.2.14/263
722	GPIO Pin Data Output Register (SIU_GPDO290)	8	R/W	00h	8.2.14/263
723	GPIO Pin Data Output Register (SIU_GPDO291)	8	R/W	00h	8.2.14/263
724	GPIO Pin Data Output Register (SIU_GPDO292)	8	R/W	00h	8.2.14/263
725	GPIO Pin Data Output Register (SIU_GPDO293)	8	R/W	00h	8.2.14/263
726	GPIO Pin Data Output Register (SIU_GPDO294)	8	R/W	00h	8.2.14/263
727	GPIO Pin Data Output Register (SIU_GPDO295)	8	R/W	00h	8.2.14/263
728	GPIO Pin Data Output Register (SIU_GPDO296)	8	R/W	00h	8.2.14/263
729	GPIO Pin Data Output Register (SIU_GPDO297)	8	R/W	00h	8.2.14/263
72A	GPIO Pin Data Output Register (SIU_GPDO298)	8	R/W	00h	8.2.14/263
72B	GPIO Pin Data Output Register (SIU_GPDO299)	8	R/W	00h	8.2.14/263
72C	GPIO Pin Data Output Register (SIU_GPDO300)	8	R/W	00h	8.2.14/263
72D	GPIO Pin Data Output Register (SIU_GPDO301)	8	R/W	00h	8.2.14/263
72E	GPIO Pin Data Output Register (SIU_GPDO302)	8	R/W	00h	8.2.14/263
72F	GPIO Pin Data Output Register (SIU_GPDO303)	8	R/W	00h	8.2.14/263

Table continues on the next page...

SIU memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
730	GPIO Pin Data Output Register (SIU_GPDO304)	8	R/W	00h	8.2.14/263
731	GPIO Pin Data Output Register (SIU_GPDO305)	8	R/W	00h	8.2.14/263
732	GPIO Pin Data Output Register (SIU_GPDO306)	8	R/W	00h	8.2.14/263
733	GPIO Pin Data Output Register (SIU_GPDO307)	8	R/W	00h	8.2.14/263
734	GPIO Pin Data Output Register (SIU_GPDO308)	8	R/W	00h	8.2.14/263
735	GPIO Pin Data Output Register (SIU_GPDO309)	8	R/W	00h	8.2.14/263
736	GPIO Pin Data Output Register (SIU_GPDO310)	8	R/W	00h	8.2.14/263
737	GPIO Pin Data Output Register (SIU_GPDO311)	8	R/W	00h	8.2.14/263
738	GPIO Pin Data Output Register (SIU_GPDO312)	8	R/W	00h	8.2.14/263
739	GPIO Pin Data Output Register (SIU_GPDO313)	8	R/W	00h	8.2.14/263
73A	GPIO Pin Data Output Register (SIU_GPDO314)	8	R/W	00h	8.2.14/263
73B	GPIO Pin Data Output Register (SIU_GPDO315)	8	R/W	00h	8.2.14/263
73C	GPIO Pin Data Output Register (SIU_GPDO316)	8	R/W	00h	8.2.14/263
73D	GPIO Pin Data Output Register (SIU_GPDO317)	8	R/W	00h	8.2.14/263
73E	GPIO Pin Data Output Register (SIU_GPDO318)	8	R/W	00h	8.2.14/263
73F	GPIO Pin Data Output Register (SIU_GPDO319)	8	R/W	00h	8.2.14/263
740	GPIO Pin Data Output Register (SIU_GPDO320)	8	R/W	00h	8.2.14/263
741	GPIO Pin Data Output Register (SIU_GPDO321)	8	R/W	00h	8.2.14/263
742	GPIO Pin Data Output Register (SIU_GPDO322)	8	R/W	00h	8.2.14/263
743	GPIO Pin Data Output Register (SIU_GPDO323)	8	R/W	00h	8.2.14/263
744	GPIO Pin Data Output Register (SIU_GPDO324)	8	R/W	00h	8.2.14/263
745	GPIO Pin Data Output Register (SIU_GPDO325)	8	R/W	00h	8.2.14/263
746	GPIO Pin Data Output Register (SIU_GPDO326)	8	R/W	00h	8.2.14/263
747	GPIO Pin Data Output Register (SIU_GPDO327)	8	R/W	00h	8.2.14/263
748	GPIO Pin Data Output Register (SIU_GPDO328)	8	R/W	00h	8.2.14/263
749	GPIO Pin Data Output Register (SIU_GPDO329)	8	R/W	00h	8.2.14/263
74A	GPIO Pin Data Output Register (SIU_GPDO330)	8	R/W	00h	8.2.14/263
74B	GPIO Pin Data Output Register (SIU_GPDO331)	8	R/W	00h	8.2.14/263
74C	GPIO Pin Data Output Register (SIU_GPDO332)	8	R/W	00h	8.2.14/263
74D	GPIO Pin Data Output Register (SIU_GPDO333)	8	R/W	00h	8.2.14/263
74E	GPIO Pin Data Output Register (SIU_GPDO334)	8	R/W	00h	8.2.14/263
74F	GPIO Pin Data Output Register (SIU_GPDO335)	8	R/W	00h	8.2.14/263
750	GPIO Pin Data Output Register (SIU_GPDO336)	8	R/W	00h	8.2.14/263
751	GPIO Pin Data Output Register (SIU_GPDO337)	8	R/W	00h	8.2.14/263
752	GPIO Pin Data Output Register (SIU_GPDO338)	8	R/W	00h	8.2.14/263
753	GPIO Pin Data Output Register (SIU_GPDO339)	8	R/W	00h	8.2.14/263
754	GPIO Pin Data Output Register (SIU_GPDO340)	8	R/W	00h	8.2.14/263
755	GPIO Pin Data Output Register (SIU_GPDO341)	8	R/W	00h	8.2.14/263
756	GPIO Pin Data Output Register (SIU_GPDO342)	8	R/W	00h	8.2.14/263

Table continues on the next page...

SIU memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
757	GPIO Pin Data Output Register (SIU_GPDO343)	8	R/W	00h	8.2.14/263
758	GPIO Pin Data Output Register (SIU_GPDO344)	8	R/W	00h	8.2.14/263
759	GPIO Pin Data Output Register (SIU_GPDO345)	8	R/W	00h	8.2.14/263
75A	GPIO Pin Data Output Register (SIU_GPDO346)	8	R/W	00h	8.2.14/263
75B	GPIO Pin Data Output Register (SIU_GPDO347)	8	R/W	00h	8.2.14/263
75C	GPIO Pin Data Output Register (SIU_GPDO348)	8	R/W	00h	8.2.14/263
75D	GPIO Pin Data Output Register (SIU_GPDO349)	8	R/W	00h	8.2.14/263
75E	GPIO Pin Data Output Register (SIU_GPDO350)	8	R/W	00h	8.2.14/263
75F	GPIO Pin Data Output Register (SIU_GPDO351)	8	R/W	00h	8.2.14/263
760	GPIO Pin Data Output Register (SIU_GPDO352)	8	R/W	00h	8.2.14/263
761	GPIO Pin Data Output Register (SIU_GPDO353)	8	R/W	00h	8.2.14/263
762	GPIO Pin Data Output Register (SIU_GPDO354)	8	R/W	00h	8.2.14/263
763	GPIO Pin Data Output Register (SIU_GPDO355)	8	R/W	00h	8.2.14/263
764	GPIO Pin Data Output Register (SIU_GPDO356)	8	R/W	00h	8.2.14/263
765	GPIO Pin Data Output Register (SIU_GPDO357)	8	R/W	00h	8.2.14/263
766	GPIO Pin Data Output Register (SIU_GPDO358)	8	R/W	00h	8.2.14/263
767	GPIO Pin Data Output Register (SIU_GPDO359)	8	R/W	00h	8.2.14/263
768	GPIO Pin Data Output Register (SIU_GPDO360)	8	R/W	00h	8.2.14/263
769	GPIO Pin Data Output Register (SIU_GPDO361)	8	R/W	00h	8.2.14/263
76A	GPIO Pin Data Output Register (SIU_GPDO362)	8	R/W	00h	8.2.14/263
76B	GPIO Pin Data Output Register (SIU_GPDO363)	8	R/W	00h	8.2.14/263
76C	GPIO Pin Data Output Register (SIU_GPDO364)	8	R/W	00h	8.2.14/263
76D	GPIO Pin Data Output Register (SIU_GPDO365)	8	R/W	00h	8.2.14/263
76E	GPIO Pin Data Output Register (SIU_GPDO366)	8	R/W	00h	8.2.14/263
76F	GPIO Pin Data Output Register (SIU_GPDO367)	8	R/W	00h	8.2.14/263
770	GPIO Pin Data Output Register (SIU_GPDO368)	8	R/W	00h	8.2.14/263
771	GPIO Pin Data Output Register (SIU_GPDO369)	8	R/W	00h	8.2.14/263
772	GPIO Pin Data Output Register (SIU_GPDO370)	8	R/W	00h	8.2.14/263
773	GPIO Pin Data Output Register (SIU_GPDO371)	8	R/W	00h	8.2.14/263
774	GPIO Pin Data Output Register (SIU_GPDO372)	8	R/W	00h	8.2.14/263
775	GPIO Pin Data Output Register (SIU_GPDO373)	8	R/W	00h	8.2.14/263
776	GPIO Pin Data Output Register (SIU_GPDO374)	8	R/W	00h	8.2.14/263
777	GPIO Pin Data Output Register (SIU_GPDO375)	8	R/W	00h	8.2.14/263
778	GPIO Pin Data Output Register (SIU_GPDO376)	8	R/W	00h	8.2.14/263
779	GPIO Pin Data Output Register (SIU_GPDO377)	8	R/W	00h	8.2.14/263
77A	GPIO Pin Data Output Register (SIU_GPDO378)	8	R/W	00h	8.2.14/263
77B	GPIO Pin Data Output Register (SIU_GPDO379)	8	R/W	00h	8.2.14/263
77C	GPIO Pin Data Output Register (SIU_GPDO380)	8	R/W	00h	8.2.14/263
77D	GPIO Pin Data Output Register (SIU_GPDO381)	8	R/W	00h	8.2.14/263

Table continues on the next page...

SIU memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
77E	GPIO Pin Data Output Register (SIU_GPDO382)	8	R/W	00h	8.2.14/263
77F	GPIO Pin Data Output Register (SIU_GPDO383)	8	R/W	00h	8.2.14/263
780	GPIO Pin Data Output Register (SIU_GPDO384)	8	R/W	00h	8.2.14/263
781	GPIO Pin Data Output Register (SIU_GPDO385)	8	R/W	00h	8.2.14/263
782	GPIO Pin Data Output Register (SIU_GPDO386)	8	R/W	00h	8.2.14/263
783	GPIO Pin Data Output Register (SIU_GPDO387)	8	R/W	00h	8.2.14/263
784	GPIO Pin Data Output Register (SIU_GPDO388)	8	R/W	00h	8.2.14/263
785	GPIO Pin Data Output Register (SIU_GPDO389)	8	R/W	00h	8.2.14/263
786	GPIO Pin Data Output Register (SIU_GPDO390)	8	R/W	00h	8.2.14/263
787	GPIO Pin Data Output Register (SIU_GPDO391)	8	R/W	00h	8.2.14/263
788	GPIO Pin Data Output Register (SIU_GPDO392)	8	R/W	00h	8.2.14/263
789	GPIO Pin Data Output Register (SIU_GPDO393)	8	R/W	00h	8.2.14/263
78A	GPIO Pin Data Output Register (SIU_GPDO394)	8	R/W	00h	8.2.14/263
78B	GPIO Pin Data Output Register (SIU_GPDO395)	8	R/W	00h	8.2.14/263
78C	GPIO Pin Data Output Register (SIU_GPDO396)	8	R/W	00h	8.2.14/263
78D	GPIO Pin Data Output Register (SIU_GPDO397)	8	R/W	00h	8.2.14/263
78E	GPIO Pin Data Output Register (SIU_GPDO398)	8	R/W	00h	8.2.14/263
78F	GPIO Pin Data Output Register (SIU_GPDO399)	8	R/W	00h	8.2.14/263
790	GPIO Pin Data Output Register (SIU_GPDO400)	8	R/W	00h	8.2.14/263
791	GPIO Pin Data Output Register (SIU_GPDO401)	8	R/W	00h	8.2.14/263
792	GPIO Pin Data Output Register (SIU_GPDO402)	8	R/W	00h	8.2.14/263
793	GPIO Pin Data Output Register (SIU_GPDO403)	8	R/W	00h	8.2.14/263
794	GPIO Pin Data Output Register (SIU_GPDO404)	8	R/W	00h	8.2.14/263
795	GPIO Pin Data Output Register (SIU_GPDO405)	8	R/W	00h	8.2.14/263
796	GPIO Pin Data Output Register (SIU_GPDO406)	8	R/W	00h	8.2.14/263
797	GPIO Pin Data Output Register (SIU_GPDO407)	8	R/W	00h	8.2.14/263
798	GPIO Pin Data Output Register (SIU_GPDO408)	8	R/W	00h	8.2.14/263
799	GPIO Pin Data Output Register (SIU_GPDO409)	8	R/W	00h	8.2.14/263
79A	GPIO Pin Data Output Register (SIU_GPDO410)	8	R/W	00h	8.2.14/263
79B	GPIO Pin Data Output Register (SIU_GPDO411)	8	R/W	00h	8.2.14/263
79C	GPIO Pin Data Output Register (SIU_GPDO412)	8	R/W	00h	8.2.14/263
79D	GPIO Pin Data Output Register (SIU_GPDO413)	8	R/W	00h	8.2.14/263
79E	GPIO Pin Data Output Register (SIU_GPDO414)	8	R/W	00h	8.2.14/263
79F	GPIO Pin Data Output Register (SIU_GPDO415)	8	R/W	00h	8.2.14/263
7A0	GPIO Pin Data Output Register (SIU_GPDO416)	8	R/W	00h	8.2.14/263
7A1	GPIO Pin Data Output Register (SIU_GPDO417)	8	R/W	00h	8.2.14/263
7A2	GPIO Pin Data Output Register (SIU_GPDO418)	8	R/W	00h	8.2.14/263
7A3	GPIO Pin Data Output Register (SIU_GPDO419)	8	R/W	00h	8.2.14/263
7A4	GPIO Pin Data Output Register (SIU_GPDO420)	8	R/W	00h	8.2.14/263

Table continues on the next page...

SIU memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
7A5	GPIO Pin Data Output Register (SIU_GPDO421)	8	R/W	00h	8.2.14/263
7A6	GPIO Pin Data Output Register (SIU_GPDO422)	8	R/W	00h	8.2.14/263
7A7	GPIO Pin Data Output Register (SIU_GPDO423)	8	R/W	00h	8.2.14/263
7A8	GPIO Pin Data Output Register (SIU_GPDO424)	8	R/W	00h	8.2.14/263
7A9	GPIO Pin Data Output Register (SIU_GPDO425)	8	R/W	00h	8.2.14/263
7AA	GPIO Pin Data Output Register (SIU_GPDO426)	8	R/W	00h	8.2.14/263
7AB	GPIO Pin Data Output Register (SIU_GPDO427)	8	R/W	00h	8.2.14/263
7AC	GPIO Pin Data Output Register (SIU_GPDO428)	8	R/W	00h	8.2.14/263
7AD	GPIO Pin Data Output Register (SIU_GPDO429)	8	R/W	00h	8.2.14/263
7AE	GPIO Pin Data Output Register (SIU_GPDO430)	8	R/W	00h	8.2.14/263
7AF	GPIO Pin Data Output Register (SIU_GPDO431)	8	R/W	00h	8.2.14/263
7B0	GPIO Pin Data Output Register (SIU_GPDO432)	8	R/W	00h	8.2.14/263
7B1	GPIO Pin Data Output Register (SIU_GPDO433)	8	R/W	00h	8.2.14/263
7B2	GPIO Pin Data Output Register (SIU_GPDO434)	8	R/W	00h	8.2.14/263
7B3	GPIO Pin Data Output Register (SIU_GPDO435)	8	R/W	00h	8.2.14/263
7B4	GPIO Pin Data Output Register (SIU_GPDO436)	8	R/W	00h	8.2.14/263
7B5	GPIO Pin Data Output Register (SIU_GPDO437)	8	R/W	00h	8.2.14/263
7B6	GPIO Pin Data Output Register (SIU_GPDO438)	8	R/W	00h	8.2.14/263
7B7	GPIO Pin Data Output Register (SIU_GPDO439)	8	R/W	00h	8.2.14/263
7B8	GPIO Pin Data Output Register (SIU_GPDO440)	8	R/W	00h	8.2.14/263
7B9	GPIO Pin Data Output Register (SIU_GPDO441)	8	R/W	00h	8.2.14/263
7BA	GPIO Pin Data Output Register (SIU_GPDO442)	8	R/W	00h	8.2.14/263
7BB	GPIO Pin Data Output Register (SIU_GPDO443)	8	R/W	00h	8.2.14/263
7BC	GPIO Pin Data Output Register (SIU_GPDO444)	8	R/W	00h	8.2.14/263
7BD	GPIO Pin Data Output Register (SIU_GPDO445)	8	R/W	00h	8.2.14/263
7BE	GPIO Pin Data Output Register (SIU_GPDO446)	8	R/W	00h	8.2.14/263
7BF	GPIO Pin Data Output Register (SIU_GPDO447)	8	R/W	00h	8.2.14/263
7C0	GPIO Pin Data Output Register (SIU_GPDO448)	8	R/W	00h	8.2.14/263
7C1	GPIO Pin Data Output Register (SIU_GPDO449)	8	R/W	00h	8.2.14/263
7C2	GPIO Pin Data Output Register (SIU_GPDO450)	8	R/W	00h	8.2.14/263
7C3	GPIO Pin Data Output Register (SIU_GPDO451)	8	R/W	00h	8.2.14/263
7C4	GPIO Pin Data Output Register (SIU_GPDO452)	8	R/W	00h	8.2.14/263
7C5	GPIO Pin Data Output Register (SIU_GPDO453)	8	R/W	00h	8.2.14/263
7C6	GPIO Pin Data Output Register (SIU_GPDO454)	8	R/W	00h	8.2.14/263
7C7	GPIO Pin Data Output Register (SIU_GPDO455)	8	R/W	00h	8.2.14/263
7C8	GPIO Pin Data Output Register (SIU_GPDO456)	8	R/W	00h	8.2.14/263
7C9	GPIO Pin Data Output Register (SIU_GPDO457)	8	R/W	00h	8.2.14/263
7CA	GPIO Pin Data Output Register (SIU_GPDO458)	8	R/W	00h	8.2.14/263
7CB	GPIO Pin Data Output Register (SIU_GPDO459)	8	R/W	00h	8.2.14/263

Table continues on the next page...

SIU memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
7CC	GPIO Pin Data Output Register (SIU_GPDO460)	8	R/W	00h	8.2.14/263
7CD	GPIO Pin Data Output Register (SIU_GPDO461)	8	R/W	00h	8.2.14/263
7CE	GPIO Pin Data Output Register (SIU_GPDO462)	8	R/W	00h	8.2.14/263
7CF	GPIO Pin Data Output Register (SIU_GPDO463)	8	R/W	00h	8.2.14/263
7D0	GPIO Pin Data Output Register (SIU_GPDO464)	8	R/W	00h	8.2.14/263
7D1	GPIO Pin Data Output Register (SIU_GPDO465)	8	R/W	00h	8.2.14/263
7D2	GPIO Pin Data Output Register (SIU_GPDO466)	8	R/W	00h	8.2.14/263
7D3	GPIO Pin Data Output Register (SIU_GPDO467)	8	R/W	00h	8.2.14/263
7D4	GPIO Pin Data Output Register (SIU_GPDO468)	8	R/W	00h	8.2.14/263
7D5	GPIO Pin Data Output Register (SIU_GPDO469)	8	R/W	00h	8.2.14/263
7D6	GPIO Pin Data Output Register (SIU_GPDO470)	8	R/W	00h	8.2.14/263
7D7	GPIO Pin Data Output Register (SIU_GPDO471)	8	R/W	00h	8.2.14/263
7D8	GPIO Pin Data Output Register (SIU_GPDO472)	8	R/W	00h	8.2.14/263
7D9	GPIO Pin Data Output Register (SIU_GPDO473)	8	R/W	00h	8.2.14/263
7DA	GPIO Pin Data Output Register (SIU_GPDO474)	8	R/W	00h	8.2.14/263
7DB	GPIO Pin Data Output Register (SIU_GPDO475)	8	R/W	00h	8.2.14/263
7DC	GPIO Pin Data Output Register (SIU_GPDO476)	8	R/W	00h	8.2.14/263
7DD	GPIO Pin Data Output Register (SIU_GPDO477)	8	R/W	00h	8.2.14/263
7DE	GPIO Pin Data Output Register (SIU_GPDO478)	8	R/W	00h	8.2.14/263
7DF	GPIO Pin Data Output Register (SIU_GPDO479)	8	R/W	00h	8.2.14/263
7E0	GPIO Pin Data Output Register (SIU_GPDO480)	8	R/W	00h	8.2.14/263
7E1	GPIO Pin Data Output Register (SIU_GPDO481)	8	R/W	00h	8.2.14/263
7E2	GPIO Pin Data Output Register (SIU_GPDO482)	8	R/W	00h	8.2.14/263
7E3	GPIO Pin Data Output Register (SIU_GPDO483)	8	R/W	00h	8.2.14/263
7E4	GPIO Pin Data Output Register (SIU_GPDO484)	8	R/W	00h	8.2.14/263
7E5	GPIO Pin Data Output Register (SIU_GPDO485)	8	R/W	00h	8.2.14/263
7E6	GPIO Pin Data Output Register (SIU_GPDO486)	8	R/W	00h	8.2.14/263
7E7	GPIO Pin Data Output Register (SIU_GPDO487)	8	R/W	00h	8.2.14/263
7E8	GPIO Pin Data Output Register (SIU_GPDO488)	8	R/W	00h	8.2.14/263
7E9	GPIO Pin Data Output Register (SIU_GPDO489)	8	R/W	00h	8.2.14/263
7EA	GPIO Pin Data Output Register (SIU_GPDO490)	8	R/W	00h	8.2.14/263
7EB	GPIO Pin Data Output Register (SIU_GPDO491)	8	R/W	00h	8.2.14/263
84B	GPIO Pin Data Input Register (legacy) (SIU_GPDI75L)	8	R	00h	8.2.15/264
84C	GPIO Pin Data Input Register (legacy) (SIU_GPDI76L)	8	R	00h	8.2.15/264
84D	GPIO Pin Data Input Register (legacy) (SIU_GPDI77L)	8	R	00h	8.2.15/264
84E	GPIO Pin Data Input Register (legacy) (SIU_GPDI78L)	8	R	00h	8.2.15/264
84F	GPIO Pin Data Input Register (legacy) (SIU_GPDI79L)	8	R	00h	8.2.15/264
850	GPIO Pin Data Input Register (legacy) (SIU_GPDI80L)	8	R	00h	8.2.15/264
851	GPIO Pin Data Input Register (legacy) (SIU_GPDI81L)	8	R	00h	8.2.15/264

Table continues on the next page...

SIU memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
852	GPIO Pin Data Input Register (legacy) (SIU_GPDI82L)	8	R	00h	8.2.15/264
853	GPIO Pin Data Input Register (legacy) (SIU_GPDI83L)	8	R	00h	8.2.15/264
854	GPIO Pin Data Input Register (legacy) (SIU_GPDI84L)	8	R	00h	8.2.15/264
855	GPIO Pin Data Input Register (legacy) (SIU_GPDI85L)	8	R	00h	8.2.15/264
856	GPIO Pin Data Input Register (legacy) (SIU_GPDI86L)	8	R	00h	8.2.15/264
857	GPIO Pin Data Input Register (legacy) (SIU_GPDI87L)	8	R	00h	8.2.15/264
858	GPIO Pin Data Input Register (legacy) (SIU_GPDI88L)	8	R	00h	8.2.15/264
859	GPIO Pin Data Input Register (legacy) (SIU_GPDI89L)	8	R	00h	8.2.15/264
85A	GPIO Pin Data Input Register (legacy) (SIU_GPDI90L)	8	R	00h	8.2.15/264
85B	GPIO Pin Data Input Register (legacy) (SIU_GPDI91L)	8	R	00h	8.2.15/264
85C	GPIO Pin Data Input Register (legacy) (SIU_GPDI92L)	8	R	00h	8.2.15/264
85D	GPIO Pin Data Input Register (legacy) (SIU_GPDI93L)	8	R	00h	8.2.15/264
85E	GPIO Pin Data Input Register (legacy) (SIU_GPDI94L)	8	R	00h	8.2.15/264
85F	GPIO Pin Data Input Register (legacy) (SIU_GPDI95L)	8	R	00h	8.2.15/264
860	GPIO Pin Data Input Register (legacy) (SIU_GPDI96L)	8	R	00h	8.2.15/264
861	GPIO Pin Data Input Register (legacy) (SIU_GPDI97L)	8	R	00h	8.2.15/264
862	GPIO Pin Data Input Register (legacy) (SIU_GPDI98L)	8	R	00h	8.2.15/264
863	GPIO Pin Data Input Register (legacy) (SIU_GPDI99L)	8	R	00h	8.2.15/264
864	GPIO Pin Data Input Register (legacy) (SIU_GPDI100L)	8	R	00h	8.2.15/264
865	GPIO Pin Data Input Register (legacy) (SIU_GPDI101L)	8	R	00h	8.2.15/264
866	GPIO Pin Data Input Register (legacy) (SIU_GPDI102L)	8	R	00h	8.2.15/264
867	GPIO Pin Data Input Register (legacy) (SIU_GPDI103L)	8	R	00h	8.2.15/264
868	GPIO Pin Data Input Register (legacy) (SIU_GPDI104L)	8	R	00h	8.2.15/264
869	GPIO Pin Data Input Register (legacy) (SIU_GPDI105L)	8	R	00h	8.2.15/264
86A	GPIO Pin Data Input Register (legacy) (SIU_GPDI106L)	8	R	00h	8.2.15/264
86B	GPIO Pin Data Input Register (legacy) (SIU_GPDI107L)	8	R	00h	8.2.15/264
86C	GPIO Pin Data Input Register (legacy) (SIU_GPDI108L)	8	R	00h	8.2.15/264
86D	GPIO Pin Data Input Register (legacy) (SIU_GPDI109L)	8	R	00h	8.2.15/264
86E	GPIO Pin Data Input Register (legacy) (SIU_GPDI110L)	8	R	00h	8.2.15/264
86F	GPIO Pin Data Input Register (legacy) (SIU_GPDI111L)	8	R	00h	8.2.15/264
870	GPIO Pin Data Input Register (legacy) (SIU_GPDI112L)	8	R	00h	8.2.15/264
871	GPIO Pin Data Input Register (legacy) (SIU_GPDI113L)	8	R	00h	8.2.15/264
872	GPIO Pin Data Input Register (legacy) (SIU_GPDI114L)	8	R	00h	8.2.15/264
873	GPIO Pin Data Input Register (legacy) (SIU_GPDI115L)	8	R	00h	8.2.15/264
874	GPIO Pin Data Input Register (legacy) (SIU_GPDI116L)	8	R	00h	8.2.15/264
875	GPIO Pin Data Input Register (legacy) (SIU_GPDI117L)	8	R	00h	8.2.15/264
876	GPIO Pin Data Input Register (legacy) (SIU_GPDI118L)	8	R	00h	8.2.15/264
877	GPIO Pin Data Input Register (legacy) (SIU_GPDI119L)	8	R	00h	8.2.15/264
878	GPIO Pin Data Input Register (legacy) (SIU_GPDI120L)	8	R	00h	8.2.15/264

Table continues on the next page...

SIU memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
879	GPIO Pin Data Input Register (legacy) (SIU_GPDI121L)	8	R	00h	8.2.15/264
87A	GPIO Pin Data Input Register (legacy) (SIU_GPDI122L)	8	R	00h	8.2.15/264
87B	GPIO Pin Data Input Register (legacy) (SIU_GPDI123L)	8	R	00h	8.2.15/264
87C	GPIO Pin Data Input Register (legacy) (SIU_GPDI124L)	8	R	00h	8.2.15/264
87D	GPIO Pin Data Input Register (legacy) (SIU_GPDI125L)	8	R	00h	8.2.15/264
87E	GPIO Pin Data Input Register (legacy) (SIU_GPDI126L)	8	R	00h	8.2.15/264
87F	GPIO Pin Data Input Register (legacy) (SIU_GPDI127L)	8	R	00h	8.2.15/264
880	GPIO Pin Data Input Register (legacy) (SIU_GPDI128L)	8	R	00h	8.2.15/264
881	GPIO Pin Data Input Register (legacy) (SIU_GPDI129L)	8	R	00h	8.2.15/264
882	GPIO Pin Data Input Register (legacy) (SIU_GPDI130L)	8	R	00h	8.2.15/264
883	GPIO Pin Data Input Register (legacy) (SIU_GPDI131L)	8	R	00h	8.2.15/264
884	GPIO Pin Data Input Register (legacy) (SIU_GPDI132L)	8	R	00h	8.2.15/264
885	GPIO Pin Data Input Register (legacy) (SIU_GPDI133L)	8	R	00h	8.2.15/264
886	GPIO Pin Data Input Register (legacy) (SIU_GPDI134L)	8	R	00h	8.2.15/264
887	GPIO Pin Data Input Register (legacy) (SIU_GPDI135L)	8	R	00h	8.2.15/264
888	GPIO Pin Data Input Register (legacy) (SIU_GPDI136L)	8	R	00h	8.2.15/264
889	GPIO Pin Data Input Register (legacy) (SIU_GPDI137L)	8	R	00h	8.2.15/264
88A	GPIO Pin Data Input Register (legacy) (SIU_GPDI138L)	8	R	00h	8.2.15/264
88B	GPIO Pin Data Input Register (legacy) (SIU_GPDI139L)	8	R	00h	8.2.15/264
88C	GPIO Pin Data Input Register (legacy) (SIU_GPDI140L)	8	R	00h	8.2.15/264
88D	GPIO Pin Data Input Register (legacy) (SIU_GPDI141L)	8	R	00h	8.2.15/264
88E	GPIO Pin Data Input Register (legacy) (SIU_GPDI142L)	8	R	00h	8.2.15/264
88F	GPIO Pin Data Input Register (legacy) (SIU_GPDI143L)	8	R	00h	8.2.15/264
890	GPIO Pin Data Input Register (legacy) (SIU_GPDI144L)	8	R	00h	8.2.15/264
891	GPIO Pin Data Input Register (legacy) (SIU_GPDI145L)	8	R	00h	8.2.15/264
892	GPIO Pin Data Input Register (legacy) (SIU_GPDI146L)	8	R	00h	8.2.15/264
893	GPIO Pin Data Input Register (legacy) (SIU_GPDI147L)	8	R	00h	8.2.15/264
894	GPIO Pin Data Input Register (legacy) (SIU_GPDI148L)	8	R	00h	8.2.15/264
895	GPIO Pin Data Input Register (legacy) (SIU_GPDI149L)	8	R	00h	8.2.15/264
896	GPIO Pin Data Input Register (legacy) (SIU_GPDI150L)	8	R	00h	8.2.15/264
897	GPIO Pin Data Input Register (legacy) (SIU_GPDI151L)	8	R	00h	8.2.15/264
898	GPIO Pin Data Input Register (legacy) (SIU_GPDI152L)	8	R	00h	8.2.15/264
899	GPIO Pin Data Input Register (legacy) (SIU_GPDI153L)	8	R	00h	8.2.15/264
89A	GPIO Pin Data Input Register (legacy) (SIU_GPDI154L)	8	R	00h	8.2.15/264
89B	GPIO Pin Data Input Register (legacy) (SIU_GPDI155L)	8	R	00h	8.2.15/264
89C	GPIO Pin Data Input Register (legacy) (SIU_GPDI156L)	8	R	00h	8.2.15/264
89D	GPIO Pin Data Input Register (legacy) (SIU_GPDI157L)	8	R	00h	8.2.15/264
89E	GPIO Pin Data Input Register (legacy) (SIU_GPDI158L)	8	R	00h	8.2.15/264
89F	GPIO Pin Data Input Register (legacy) (SIU_GPDI159L)	8	R	00h	8.2.15/264

Table continues on the next page...

SIU memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
8A0	GPIO Pin Data Input Register (legacy) (SIU_GPDI160L)	8	R	00h	8.2.15/264
8A1	GPIO Pin Data Input Register (legacy) (SIU_GPDI161L)	8	R	00h	8.2.15/264
8A2	GPIO Pin Data Input Register (legacy) (SIU_GPDI162L)	8	R	00h	8.2.15/264
8A3	GPIO Pin Data Input Register (legacy) (SIU_GPDI163L)	8	R	00h	8.2.15/264
8A4	GPIO Pin Data Input Register (legacy) (SIU_GPDI164L)	8	R	00h	8.2.15/264
8A5	GPIO Pin Data Input Register (legacy) (SIU_GPDI165L)	8	R	00h	8.2.15/264
8A6	GPIO Pin Data Input Register (legacy) (SIU_GPDI166L)	8	R	00h	8.2.15/264
8A7	GPIO Pin Data Input Register (legacy) (SIU_GPDI167L)	8	R	00h	8.2.15/264
8A8	GPIO Pin Data Input Register (legacy) (SIU_GPDI168L)	8	R	00h	8.2.15/264
8A9	GPIO Pin Data Input Register (legacy) (SIU_GPDI169L)	8	R	00h	8.2.15/264
8AA	GPIO Pin Data Input Register (legacy) (SIU_GPDI170L)	8	R	00h	8.2.15/264
8AB	GPIO Pin Data Input Register (legacy) (SIU_GPDI171L)	8	R	00h	8.2.15/264
8AC	GPIO Pin Data Input Register (legacy) (SIU_GPDI172L)	8	R	00h	8.2.15/264
8AD	GPIO Pin Data Input Register (legacy) (SIU_GPDI173L)	8	R	00h	8.2.15/264
8AE	GPIO Pin Data Input Register (legacy) (SIU_GPDI174L)	8	R	00h	8.2.15/264
8AF	GPIO Pin Data Input Register (legacy) (SIU_GPDI175L)	8	R	00h	8.2.15/264
8B0	GPIO Pin Data Input Register (legacy) (SIU_GPDI176L)	8	R	00h	8.2.15/264
8B1	GPIO Pin Data Input Register (legacy) (SIU_GPDI177L)	8	R	00h	8.2.15/264
8B2	GPIO Pin Data Input Register (legacy) (SIU_GPDI178L)	8	R	00h	8.2.15/264
8B3	GPIO Pin Data Input Register (legacy) (SIU_GPDI179L)	8	R	00h	8.2.15/264
8B4	GPIO Pin Data Input Register (legacy) (SIU_GPDI180L)	8	R	00h	8.2.15/264
8B5	GPIO Pin Data Input Register (legacy) (SIU_GPDI181L)	8	R	00h	8.2.15/264
8B6	GPIO Pin Data Input Register (legacy) (SIU_GPDI182L)	8	R	00h	8.2.15/264
8B7	GPIO Pin Data Input Register (legacy) (SIU_GPDI183L)	8	R	00h	8.2.15/264
8B8	GPIO Pin Data Input Register (legacy) (SIU_GPDI184L)	8	R	00h	8.2.15/264
8B9	GPIO Pin Data Input Register (legacy) (SIU_GPDI185L)	8	R	00h	8.2.15/264
8BA	GPIO Pin Data Input Register (legacy) (SIU_GPDI186L)	8	R	00h	8.2.15/264
8BB	GPIO Pin Data Input Register (legacy) (SIU_GPDI187L)	8	R	00h	8.2.15/264
8BC	GPIO Pin Data Input Register (legacy) (SIU_GPDI188L)	8	R	00h	8.2.15/264
8BD	GPIO Pin Data Input Register (legacy) (SIU_GPDI189L)	8	R	00h	8.2.15/264
8BE	GPIO Pin Data Input Register (legacy) (SIU_GPDI190L)	8	R	00h	8.2.15/264
8BF	GPIO Pin Data Input Register (legacy) (SIU_GPDI191L)	8	R	00h	8.2.15/264
8C0	GPIO Pin Data Input Register (legacy) (SIU_GPDI192L)	8	R	00h	8.2.15/264
8C1	GPIO Pin Data Input Register (legacy) (SIU_GPDI193L)	8	R	00h	8.2.15/264
8C2	GPIO Pin Data Input Register (legacy) (SIU_GPDI194L)	8	R	00h	8.2.15/264
8C3	GPIO Pin Data Input Register (legacy) (SIU_GPDI195L)	8	R	00h	8.2.15/264
8C4	GPIO Pin Data Input Register (legacy) (SIU_GPDI196L)	8	R	00h	8.2.15/264
8C5	GPIO Pin Data Input Register (legacy) (SIU_GPDI197L)	8	R	00h	8.2.15/264
8C6	GPIO Pin Data Input Register (legacy) (SIU_GPDI198L)	8	R	00h	8.2.15/264

Table continues on the next page...

SIU memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
8C7	GPIO Pin Data Input Register (legacy) (SIU_GPDI199L)	8	R	00h	8.2.15/264
8C8	GPIO Pin Data Input Register (legacy) (SIU_GPDI200L)	8	R	00h	8.2.15/264
8C9	GPIO Pin Data Input Register (legacy) (SIU_GPDI201L)	8	R	00h	8.2.15/264
8CA	GPIO Pin Data Input Register (legacy) (SIU_GPDI202L)	8	R	00h	8.2.15/264
8CB	GPIO Pin Data Input Register (legacy) (SIU_GPDI203L)	8	R	00h	8.2.15/264
8CC	GPIO Pin Data Input Register (legacy) (SIU_GPDI204L)	8	R	00h	8.2.15/264
8CD	GPIO Pin Data Input Register (legacy) (SIU_GPDI205L)	8	R	00h	8.2.15/264
8CE	GPIO Pin Data Input Register (legacy) (SIU_GPDI206L)	8	R	00h	8.2.15/264
8CF	GPIO Pin Data Input Register (legacy) (SIU_GPDI207L)	8	R	00h	8.2.15/264
8D0	GPIO Pin Data Input Register (legacy) (SIU_GPDI208L)	8	R	00h	8.2.15/264
8D1	GPIO Pin Data Input Register (legacy) (SIU_GPDI209L)	8	R	00h	8.2.15/264
8D2	GPIO Pin Data Input Register (legacy) (SIU_GPDI210L)	8	R	00h	8.2.15/264
8D3	GPIO Pin Data Input Register (legacy) (SIU_GPDI211L)	8	R	00h	8.2.15/264
8D4	GPIO Pin Data Input Register (legacy) (SIU_GPDI212L)	8	R	00h	8.2.15/264
8D5	GPIO Pin Data Input Register (legacy) (SIU_GPDI213L)	8	R	00h	8.2.15/264
8D6	GPIO Pin Data Input Register (legacy) (SIU_GPDI214L)	8	R	00h	8.2.15/264
8D7	GPIO Pin Data Input Register (legacy) (SIU_GPDI215L)	8	R	00h	8.2.15/264
8D8	GPIO Pin Data Input Register (legacy) (SIU_GPDI216L)	8	R	00h	8.2.15/264
8D9	GPIO Pin Data Input Register (legacy) (SIU_GPDI217L)	8	R	00h	8.2.15/264
8DA	GPIO Pin Data Input Register (legacy) (SIU_GPDI218L)	8	R	00h	8.2.15/264
8DB	GPIO Pin Data Input Register (legacy) (SIU_GPDI219L)	8	R	00h	8.2.15/264
8DC	GPIO Pin Data Input Register (legacy) (SIU_GPDI220L)	8	R	00h	8.2.15/264
8DD	GPIO Pin Data Input Register (legacy) (SIU_GPDI221L)	8	R	00h	8.2.15/264
8DE	GPIO Pin Data Input Register (legacy) (SIU_GPDI222L)	8	R	00h	8.2.15/264
8DF	GPIO Pin Data Input Register (legacy) (SIU_GPDI223L)	8	R	00h	8.2.15/264
8E0	GPIO Pin Data Input Register (legacy) (SIU_GPDI224L)	8	R	00h	8.2.15/264
8E1	GPIO Pin Data Input Register (legacy) (SIU_GPDI225L)	8	R	00h	8.2.15/264
8E2	GPIO Pin Data Input Register (legacy) (SIU_GPDI226L)	8	R	00h	8.2.15/264
8E3	GPIO Pin Data Input Register (legacy) (SIU_GPDI227L)	8	R	00h	8.2.15/264
8E4	GPIO Pin Data Input Register (legacy) (SIU_GPDI228L)	8	R	00h	8.2.15/264
8E5	GPIO Pin Data Input Register (legacy) (SIU_GPDI229L)	8	R	00h	8.2.15/264
8E6	GPIO Pin Data Input Register (legacy) (SIU_GPDI230L)	8	R	00h	8.2.15/264
8E7	GPIO Pin Data Input Register (legacy) (SIU_GPDI231L)	8	R	00h	8.2.15/264
8E8	GPIO Pin Data Input Register (legacy) (SIU_GPDI232L)	8	R	00h	8.2.15/264
8E9	GPIO Pin Data Input Register (legacy) (SIU_GPDI233L)	8	R	00h	8.2.15/264
8EA	GPIO Pin Data Input Register (legacy) (SIU_GPDI234L)	8	R	00h	8.2.15/264
8EB	GPIO Pin Data Input Register (legacy) (SIU_GPDI235L)	8	R	00h	8.2.15/264
8EC	GPIO Pin Data Input Register (legacy) (SIU_GPDI236L)	8	R	00h	8.2.15/264
8ED	GPIO Pin Data Input Register (legacy) (SIU_GPDI237L)	8	R	00h	8.2.15/264

Table continues on the next page...

SIU memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
8EE	GPIO Pin Data Input Register (legacy) (SIU_GPDI238L)	8	R	00h	8.2.15/264
8EF	GPIO Pin Data Input Register (legacy) (SIU_GPDI239L)	8	R	00h	8.2.15/264
8F0	GPIO Pin Data Input Register (legacy) (SIU_GPDI240L)	8	R	00h	8.2.15/264
8F1	GPIO Pin Data Input Register (legacy) (SIU_GPDI241L)	8	R	00h	8.2.15/264
8F2	GPIO Pin Data Input Register (legacy) (SIU_GPDI242L)	8	R	00h	8.2.15/264
8F3	GPIO Pin Data Input Register (legacy) (SIU_GPDI243L)	8	R	00h	8.2.15/264
8F4	GPIO Pin Data Input Register (legacy) (SIU_GPDI244L)	8	R	00h	8.2.15/264
8F5	GPIO Pin Data Input Register (legacy) (SIU_GPDI245L)	8	R	00h	8.2.15/264
8F6	GPIO Pin Data Input Register (legacy) (SIU_GPDI246L)	8	R	00h	8.2.15/264
8F7	GPIO Pin Data Input Register (legacy) (SIU_GPDI247L)	8	R	00h	8.2.15/264
8F8	GPIO Pin Data Input Register (legacy) (SIU_GPDI248L)	8	R	00h	8.2.15/264
8F9	GPIO Pin Data Input Register (legacy) (SIU_GPDI249L)	8	R	00h	8.2.15/264
8FA	GPIO Pin Data Input Register (legacy) (SIU_GPDI250L)	8	R	00h	8.2.15/264
8FB	GPIO Pin Data Input Register (legacy) (SIU_GPDI251L)	8	R	00h	8.2.15/264
8FC	GPIO Pin Data Input Register (legacy) (SIU_GPDI252L)	8	R	00h	8.2.15/264
8FD	GPIO Pin Data Input Register (legacy) (SIU_GPDI253L)	8	R	00h	8.2.15/264
904	External IRQ Input Select Register (SIU_EIISR)	32	R/W	0000_0000h	8.2.16/265
908	DSPI Input Select Register (SIU_DISR)	32	R/W	0000_0000h	8.2.17/267
910	eQADC Command FIFO Trigger Source Registers (SIU_ISEL4)	32	R/W	0000_0000h	8.2.18/269
914	eQADC Command FIFO Trigger Source Registers (SIU_ISEL5)	32	R/W	0000_0000h	8.2.19/274
918	eQADC Command FIFO Trigger Source Registers (SIU_ISEL6)	32	R/W	0000_0000h	8.2.20/277
91C	eQADC Command FIFO Trigger Source Registers (SIU_ISEL7)	32	R/W	0000_0000h	8.2.21/281
920	eTPU_A Input Select Register (SIU_ISEL8)	32	R/W	0000_0000h	8.2.22/284
924	eQADC Advance Trigger Source Register (SIU_ISEL9)	32	R/W	0000_0000h	8.2.23/285
928	Decimation Filter Register 1 (SIU_DECFIL1)	32	R/W	0000_0000h	8.2.24/288
92C	Decimation Filter Register 2 (SIU_DECFIL2)	32	R/W	0000_0000h	8.2.25/290
930	Decimation Filter Register 3 (SIU_DECFIL3)	32	R/W	0000_0000h	8.2.26/293
934	Decimation Filter Register 4 (SIU_DECFIL4)	32	R/W	0000_0000h	8.2.27/295
938	Decimation Filter Register 5 (SIU_DECFIL5)	32	R/W	0000_0000h	8.2.28/297
93C	Reaction Module Trigger Selection Register (SIU_REACTSR)	32	R/W	0000_0000h	8.2.29/298
940	Sigma-Delta Gate Selection Register (SIU_SDGATE_SEL)	32	R/W	0000_0000h	8.2.30/301
944	Sigma-Delta to eTPU_A Connection Configuration Register (SIU_SDETPUA)	32	R/W	0000_0000h	8.2.31/302
948	Sigma-Delta to eTPU_B Connection Configuration Register (SIU_SDETPUB)	32	R/W	0000_0000h	8.2.32/305

Table continues on the next page...

SIU memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
94C	Sigma-Delta to eTPU_C Connection Configuration Register (SIU_SDETPUC)	32	R/W	0000_0000h	8.2.33/307
950	Trigger Burst Generator Control Registers for eQADC_A (SIU_TBG_CR_A0)	32	R/W	0000_0000h	8.2.34/310
954	Trigger Burst Generator Control Registers for eQADC_A (SIU_TBG_CR_A1)	32	R/W	0000_0000h	8.2.34/310
958	Trigger Burst Generator Control Registers for eQADC_A (SIU_TBG_CR_A2)	32	R/W	0000_0000h	8.2.34/310
95C	Trigger Burst Generator Control Registers for eQADC_A (SIU_TBG_CR_A3)	32	R/W	0000_0000h	8.2.34/310
960	Trigger Burst Generator Control Registers for eQADC_A (SIU_TBG_CR_A4)	32	R/W	0000_0000h	8.2.34/310
964	Trigger Burst Generator Control Registers for eQADC_A (SIU_TBG_CR_A5)	32	R/W	0000_0000h	8.2.34/310
968	Trigger Burst Generator Control Registers for eQADC_B (SIU_TBG_CR_B0)	32	R/W	0000_0000h	8.2.35/311
96C	Trigger Burst Generator Control Registers for eQADC_B (SIU_TBG_CR_B1)	32	R/W	0000_0000h	8.2.35/311
970	Trigger Burst Generator Control Registers for eQADC_B (SIU_TBG_CR_B2)	32	R/W	0000_0000h	8.2.35/311
974	Trigger Burst Generator Control Registers for eQADC_B (SIU_TBG_CR_B3)	32	R/W	0000_0000h	8.2.35/311
978	Trigger Burst Generator Control Registers for eQADC_B (SIU_TBG_CR_B4)	32	R/W	0000_0000h	8.2.35/311
97C	Trigger Burst Generator Control Registers for eQADC_B (SIU_TBG_CR_B5)	32	R/W	0000_0000h	8.2.35/311
984	External Clock Control Register (SIU_ECCR)	32	R/W	0000_1001h	8.2.36/312
9A0	System Clock Register (SIU_SYSDIV)	32	R/W	0000_0010h	8.2.37/313
9A4	Halt Register 1 (SIU_HLT1)	32	R/W	4000_0000h	8.2.38/316
9A8	Halt Acknowledge Register 1 (SIU_HLTACK1)	32	R	0000_0000h	8.2.39/321
9AC	Core0 Reset Vector Register (SIU_RSTVEC0)	32	R/W	FFFF_FFFCh	8.2.40/324
9B0	Core1 Reset Vector Register (SIU_RSTVEC1)	32	R/W	FFFF_FFFEh	8.2.41/325
9B4	Core0 PID mapping control register (SIU_C0PID)	32	R/W	0000_0000h	8.2.42/327
9B8	Core1 PID mapping control register (SIU_C1PID)	32	R/W	0000_0000h	8.2.43/328
9BC	Halt Register 2 (SIU_HLT2)	32	R/W	0000_0000h	8.2.44/329
9C0	Halt Acknowledge Register 2 (SIU_HLTACK2)	32	R	0000_0000h	8.2.45/332
9C4	Sigma-Delta Clock Configuration Register (SIU_SDCLKCFG)	32	R/W	0000_0000h	8.2.46/334
9C8	LFAST Clock Configuration Register (SIU_LFCLKCFG)	32	R/W	0000_0000h	8.2.47/335
9CC	PSI5 Clock Configuration Register (SIU_PSCLKCFG)	32	R/W	0000_0000h	8.2.48/336
9D0	Reset Configuration Register (SIU_RCR)	32	R/W	See section	8.2.49/337
9D4	Lock-Step Status Register (SIU_LOCKSTEP)	32	R	0000_0000h	8.2.50/337

Table continues on the next page...

SIU memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
9D8	Progressive Clock Switch Interrupt Enable Register (SIU_PCSEIR)	32	R/W	0000_0000h	8.2.51/338
9DC	Progressive Clock Switch Interrupt Flag Register (SIU_PCSIFR)	32	R/W	0000_0000h	8.2.52/339
9E0	Fast Ethernet Controller Control Register (SIU_FECCR)	32	R/W	0000_0000h	8.2.53/340
9E4	ECC Error Injection Register (SIU_ECCEIR)	32	R/W	0000_0000h	8.2.54/340
C00	Parallel GPIO Pin Data Output Registers (SIU_PGPDO0)	32	R/W	0000_0000h	8.2.55/341
C04	Parallel GPIO Pin Data Output Registers (SIU_PGPDO1)	32	R/W	0000_0000h	8.2.55/341
C08	Parallel GPIO Pin Data Output Registers (SIU_PGPDO2)	32	R/W	0000_0000h	8.2.55/341
C0C	Parallel GPIO Pin Data Output Registers (SIU_PGPDO3)	32	R/W	0000_0000h	8.2.55/341
C10	Parallel GPIO Pin Data Output Registers (SIU_PGPDO4)	32	R/W	0000_0000h	8.2.55/341
C14	Parallel GPIO Pin Data Output Registers (SIU_PGPDO5)	32	R/W	0000_0000h	8.2.55/341
C18	Parallel GPIO Pin Data Output Registers (SIU_PGPDO6)	32	R/W	0000_0000h	8.2.55/341
C1C	Parallel GPIO Pin Data Output Registers (SIU_PGPDO7)	32	R/W	0000_0000h	8.2.55/341
C20	Parallel GPIO Pin Data Output Registers (SIU_PGPDO8)	32	R/W	0000_0000h	8.2.55/341
C24	Parallel GPIO Pin Data Output Registers (SIU_PGPDO9)	32	R/W	0000_0000h	8.2.55/341
C28	Parallel GPIO Pin Data Output Registers (SIU_PGPDO10)	32	R/W	0000_0000h	8.2.55/341
C2C	Parallel GPIO Pin Data Output Registers (SIU_PGPDO11)	32	R/W	0000_0000h	8.2.55/341
C30	Parallel GPIO Pin Data Output Registers (SIU_PGPDO12)	32	R/W	0000_0000h	8.2.55/341
C34	Parallel GPIO Pin Data Output Registers (SIU_PGPDO13)	32	R/W	0000_0000h	8.2.55/341
C38	Parallel GPIO Pin Data Output Registers (SIU_PGPDO14)	32	R/W	0000_0000h	8.2.55/341
C3C	Parallel GPIO Pin Data Output Registers (SIU_PGPDO15)	32	R/W	0000_0000h	8.2.55/341
C40	Parallel GPIO Pin Data Input Registers (SIU_PGPDI0)	32	R	0000_0000h	8.2.56/345
C44	Parallel GPIO Pin Data Input Registers (SIU_PGPDI1)	32	R	0000_0000h	8.2.56/345
C48	Parallel GPIO Pin Data Input Registers (SIU_PGPDI2)	32	R	0000_0000h	8.2.56/345
C4C	Parallel GPIO Pin Data Input Registers (SIU_PGPDI3)	32	R	0000_0000h	8.2.56/345
C50	Parallel GPIO Pin Data Input Registers (SIU_PGPDI4)	32	R	0000_0000h	8.2.56/345
C54	Parallel GPIO Pin Data Input Registers (SIU_PGPDI5)	32	R	0000_0000h	8.2.56/345
C58	Parallel GPIO Pin Data Input Registers (SIU_PGPDI6)	32	R	0000_0000h	8.2.56/345
C5C	Parallel GPIO Pin Data Input Registers (SIU_PGPDI7)	32	R	0000_0000h	8.2.56/345
C60	Parallel GPIO Pin Data Input Registers (SIU_PGPDI8)	32	R	0000_0000h	8.2.56/345
C64	Parallel GPIO Pin Data Input Registers (SIU_PGPDI9)	32	R	0000_0000h	8.2.56/345
C68	Parallel GPIO Pin Data Input Registers (SIU_PGPDI10)	32	R	0000_0000h	8.2.56/345
C6C	Parallel GPIO Pin Data Input Registers (SIU_PGPDI11)	32	R	0000_0000h	8.2.56/345
C70	Parallel GPIO Pin Data Input Registers (SIU_PGPDI12)	32	R	0000_0000h	8.2.56/345
C74	Parallel GPIO Pin Data Input Registers (SIU_PGPDI13)	32	R	0000_0000h	8.2.56/345
C78	Parallel GPIO Pin Data Input Registers (SIU_PGPDI14)	32	R	0000_0000h	8.2.56/345
C7C	Parallel GPIO Pin Data Input Registers (SIU_PGPDI15)	32	R	0000_0000h	8.2.56/345

Table continues on the next page...

SIU memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
C80	Masked Parallel GPIO Pin Data Output Registers (SIU_MPGPDO0)	32	W	0000_0000h	8.2.57/349
C84	Masked Parallel GPIO Pin Data Output Registers (SIU_MPGPDO1)	32	W	0000_0000h	8.2.57/349
C88	Masked Parallel GPIO Pin Data Output Registers (SIU_MPGPDO2)	32	W	0000_0000h	8.2.57/349
C8C	Masked Parallel GPIO Pin Data Output Registers (SIU_MPGPDO3)	32	W	0000_0000h	8.2.57/349
C90	Masked Parallel GPIO Pin Data Output Registers (SIU_MPGPDO4)	32	W	0000_0000h	8.2.57/349
C94	Masked Parallel GPIO Pin Data Output Registers (SIU_MPGPDO5)	32	W	0000_0000h	8.2.57/349
C98	Masked Parallel GPIO Pin Data Output Registers (SIU_MPGPDO6)	32	W	0000_0000h	8.2.57/349
C9C	Masked Parallel GPIO Pin Data Output Registers (SIU_MPGPDO7)	32	W	0000_0000h	8.2.57/349
CA0	Masked Parallel GPIO Pin Data Output Registers (SIU_MPGPDO8)	32	W	0000_0000h	8.2.57/349
CA4	Masked Parallel GPIO Pin Data Output Registers (SIU_MPGPDO9)	32	W	0000_0000h	8.2.57/349
CA8	Masked Parallel GPIO Pin Data Output Registers (SIU_MPGPDO10)	32	W	0000_0000h	8.2.57/349
CAC	Masked Parallel GPIO Pin Data Output Registers (SIU_MPGPDO11)	32	W	0000_0000h	8.2.57/349
CB0	Masked Parallel GPIO Pin Data Output Registers (SIU_MPGPDO12)	32	W	0000_0000h	8.2.57/349
CB4	Masked Parallel GPIO Pin Data Output Registers (SIU_MPGPDO13)	32	W	0000_0000h	8.2.57/349
CB8	Masked Parallel GPIO Pin Data Output Registers (SIU_MPGPDO14)	32	W	0000_0000h	8.2.57/349
CBC	Masked Parallel GPIO Pin Data Output Registers (SIU_MPGPDO15)	32	W	0000_0000h	8.2.57/349
CC0	Masked Parallel GPIO Pin Data Output Registers (SIU_MPGPDO16)	32	W	0000_0000h	8.2.57/349
CC4	Masked Parallel GPIO Pin Data Output Registers (SIU_MPGPDO17)	32	W	0000_0000h	8.2.57/349
CC8	Masked Parallel GPIO Pin Data Output Registers (SIU_MPGPDO18)	32	W	0000_0000h	8.2.57/349
CCC	Masked Parallel GPIO Pin Data Output Registers (SIU_MPGPDO19)	32	W	0000_0000h	8.2.57/349
CD0	Masked Parallel GPIO Pin Data Output Registers (SIU_MPGPDO20)	32	W	0000_0000h	8.2.57/349
CD4	Masked Parallel GPIO Pin Data Output Registers (SIU_MPGPDO21)	32	W	0000_0000h	8.2.57/349

Table continues on the next page...

SIU memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
CD8	Masked Parallel GPIO Pin Data Output Registers (SIU_MPGPDO22)	32	W	0000_0000h	8.2.57/349
CDC	Masked Parallel GPIO Pin Data Output Registers (SIU_MPGPDO23)	32	W	0000_0000h	8.2.57/349
CE0	Masked Parallel GPIO Pin Data Output Registers (SIU_MPGPDO24)	32	W	0000_0000h	8.2.57/349
CE4	Masked Parallel GPIO Pin Data Output Registers (SIU_MPGPDO25)	32	W	0000_0000h	8.2.57/349
CE8	Masked Parallel GPIO Pin Data Output Registers (SIU_MPGPDO26)	32	W	0000_0000h	8.2.57/349
CEC	Masked Parallel GPIO Pin Data Output Registers (SIU_MPGPDO27)	32	W	0000_0000h	8.2.57/349
CF0	Masked Parallel GPIO Pin Data Output Registers (SIU_MPGPDO28)	32	W	0000_0000h	8.2.57/349
CF4	Masked Parallel GPIO Pin Data Output Registers (SIU_MPGPDO29)	32	W	0000_0000h	8.2.57/349
CF8	Masked Parallel GPIO Pin Data Output Registers (SIU_MPGPDO30)	32	W	0000_0000h	8.2.57/349
CFC	Masked Parallel GPIO Pin Data Output Registers (SIU_MPGPDO31)	32	W	0000_0000h	8.2.57/349
D00	Mask-Output High Register (SIU_DSPIAH)	32	R/W	0000_0000h	8.2.58/353
D04	Mask-Output Low Register (SIU_DSPIAL)	32	R/W	0000_0000h	8.2.59/357
D08	Mask-Output High Register (SIU_DSPIBH)	32	R/W	0000_0000h	8.2.58/353
D0C	Mask-Output Low Register (SIU_DSPIBL)	32	R/W	0000_0000h	8.2.59/357
D10	Mask-Output High Register (SIU_DSPICH)	32	R/W	0000_0000h	8.2.58/353
D14	Mask-Output Low Register (SIU_DSPICL)	32	R/W	0000_0000h	8.2.59/357
D18	Mask-Output High Register (SIU_DSPIDH)	32	R/W	0000_0000h	8.2.58/353
D1C	Mask-Output Low Register (SIU_DSPIDL)	32	R/W	0000_0000h	8.2.59/357
D40	Serialized Output Signal Selection Register for DSPI_A (SIU_ETPUBA)	32	R/W	0000_0000h	8.2.60/362
D44	Serialized Output Signal Selection Register for DSPI_A (SIU_EMIOA)	32	R/W	0000_0000h	8.2.61/366
D48	Serialized Output Signal Selection Register for DSPI_A (SIU_DSPIAHLA)	32	R/W	0000_0000h	8.2.62/369
D50	Serialized Output Signal Selection Register for DSPI_B (SIU_ETPUAB)	32	R/W	0000_0000h	8.2.63/373
D54	Serialized Output Signal Selection Register for DSPI_B (SIU_EMIOB)	32	R/W	0000_0000h	8.2.64/377
D58	Serialized Output Signal Selection Register for DSPI_B (SIU_DSPIBHLB)	32	R/W	0000_0000h	8.2.65/381
D60	Serialized Output Signal Selection Register for DSPI_C (SIU_ETPUAC)	32	R/W	0000_0000h	8.2.66/385
D64	Serialized Output Signal Selection Register for DSPI_C (SIU_EMIOSC)	32	R/W	0000_0000h	8.2.67/388

Table continues on the next page...

SIU memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
D68	Serialized Output Signal Selection Register for DSPI_C (SIU_DSPICHLIC)	32	R/W	0000_0000h	8.2.68/392
D6C	Serialized Output Signal Selection Register for DSPI_C (SIU_ETPUBC)	32	R/W	0000_0000h	8.2.69/396
D70	Serialized Output Signal Selection Register for DSPI_D (SIU_ETPUBD)	32	R/W	0000_0000h	8.2.70/400
D74	Serialized Output Signal Selection Register for DSPI_D (SIU_EMIOSD)	32	R/W	0000_0000h	8.2.71/402
D78	Serialized Output Signal Selection Register for DSPI_D (SIU_DSPIDHLD)	32	R/W	0000_0000h	8.2.72/403
E4B	GPIO Pin Data Input Register (SIU_GPDI75)	8	R	See section	8.2.73/407
E4C	GPIO Pin Data Input Register (SIU_GPDI76)	8	R	See section	8.2.73/407
E4D	GPIO Pin Data Input Register (SIU_GPDI77)	8	R	See section	8.2.73/407
E4E	GPIO Pin Data Input Register (SIU_GPDI78)	8	R	See section	8.2.73/407
E4F	GPIO Pin Data Input Register (SIU_GPDI79)	8	R	See section	8.2.73/407
E50	GPIO Pin Data Input Register (SIU_GPDI80)	8	R	See section	8.2.73/407
E51	GPIO Pin Data Input Register (SIU_GPDI81)	8	R	See section	8.2.73/407
E52	GPIO Pin Data Input Register (SIU_GPDI82)	8	R	See section	8.2.73/407
E53	GPIO Pin Data Input Register (SIU_GPDI83)	8	R	See section	8.2.73/407
E54	GPIO Pin Data Input Register (SIU_GPDI84)	8	R	See section	8.2.73/407
E55	GPIO Pin Data Input Register (SIU_GPDI85)	8	R	See section	8.2.73/407
E56	GPIO Pin Data Input Register (SIU_GPDI86)	8	R	See section	8.2.73/407
E57	GPIO Pin Data Input Register (SIU_GPDI87)	8	R	See section	8.2.73/407
E58	GPIO Pin Data Input Register (SIU_GPDI88)	8	R	See section	8.2.73/407
E59	GPIO Pin Data Input Register (SIU_GPDI89)	8	R	See section	8.2.73/407
E5A	GPIO Pin Data Input Register (SIU_GPDI90)	8	R	See section	8.2.73/407
E5B	GPIO Pin Data Input Register (SIU_GPDI91)	8	R	See section	8.2.73/407
E5C	GPIO Pin Data Input Register (SIU_GPDI92)	8	R	See section	8.2.73/407
E5D	GPIO Pin Data Input Register (SIU_GPDI93)	8	R	See section	8.2.73/407
E5E	GPIO Pin Data Input Register (SIU_GPDI94)	8	R	See section	8.2.73/407
E5F	GPIO Pin Data Input Register (SIU_GPDI95)	8	R	See section	8.2.73/407
E60	GPIO Pin Data Input Register (SIU_GPDI96)	8	R	See section	8.2.73/407
E61	GPIO Pin Data Input Register (SIU_GPDI97)	8	R	See section	8.2.73/407
E62	GPIO Pin Data Input Register (SIU_GPDI98)	8	R	See section	8.2.73/407
E63	GPIO Pin Data Input Register (SIU_GPDI99)	8	R	See section	8.2.73/407
E64	GPIO Pin Data Input Register (SIU_GPDI100)	8	R	See section	8.2.73/407
E65	GPIO Pin Data Input Register (SIU_GPDI101)	8	R	See section	8.2.73/407
E66	GPIO Pin Data Input Register (SIU_GPDI102)	8	R	See section	8.2.73/407
E67	GPIO Pin Data Input Register (SIU_GPDI103)	8	R	See section	8.2.73/407
E68	GPIO Pin Data Input Register (SIU_GPDI104)	8	R	See section	8.2.73/407

Table continues on the next page...

SIU memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
E69	GPIO Pin Data Input Register (SIU_GPDI105)	8	R	See section	8.2.73/407
E6A	GPIO Pin Data Input Register (SIU_GPDI106)	8	R	See section	8.2.73/407
E6B	GPIO Pin Data Input Register (SIU_GPDI107)	8	R	See section	8.2.73/407
E6C	GPIO Pin Data Input Register (SIU_GPDI108)	8	R	See section	8.2.73/407
E6D	GPIO Pin Data Input Register (SIU_GPDI109)	8	R	See section	8.2.73/407
E6E	GPIO Pin Data Input Register (SIU_GPDI110)	8	R	See section	8.2.73/407
E6F	GPIO Pin Data Input Register (SIU_GPDI111)	8	R	See section	8.2.73/407
E70	GPIO Pin Data Input Register (SIU_GPDI112)	8	R	See section	8.2.73/407
E71	GPIO Pin Data Input Register (SIU_GPDI113)	8	R	See section	8.2.73/407
E72	GPIO Pin Data Input Register (SIU_GPDI114)	8	R	See section	8.2.73/407
E73	GPIO Pin Data Input Register (SIU_GPDI115)	8	R	See section	8.2.73/407
E74	GPIO Pin Data Input Register (SIU_GPDI116)	8	R	See section	8.2.73/407
E75	GPIO Pin Data Input Register (SIU_GPDI117)	8	R	See section	8.2.73/407
E76	GPIO Pin Data Input Register (SIU_GPDI118)	8	R	See section	8.2.73/407
E77	GPIO Pin Data Input Register (SIU_GPDI119)	8	R	See section	8.2.73/407
E78	GPIO Pin Data Input Register (SIU_GPDI120)	8	R	See section	8.2.73/407
E79	GPIO Pin Data Input Register (SIU_GPDI121)	8	R	See section	8.2.73/407
E7A	GPIO Pin Data Input Register (SIU_GPDI122)	8	R	See section	8.2.73/407
E7B	GPIO Pin Data Input Register (SIU_GPDI123)	8	R	See section	8.2.73/407
E7C	GPIO Pin Data Input Register (SIU_GPDI124)	8	R	See section	8.2.73/407
E7D	GPIO Pin Data Input Register (SIU_GPDI125)	8	R	See section	8.2.73/407
E7E	GPIO Pin Data Input Register (SIU_GPDI126)	8	R	See section	8.2.73/407
E7F	GPIO Pin Data Input Register (SIU_GPDI127)	8	R	See section	8.2.73/407
E80	GPIO Pin Data Input Register (SIU_GPDI128)	8	R	See section	8.2.73/407
E81	GPIO Pin Data Input Register (SIU_GPDI129)	8	R	See section	8.2.73/407
E82	GPIO Pin Data Input Register (SIU_GPDI130)	8	R	See section	8.2.73/407
E83	GPIO Pin Data Input Register (SIU_GPDI131)	8	R	See section	8.2.73/407
E84	GPIO Pin Data Input Register (SIU_GPDI132)	8	R	See section	8.2.73/407
E85	GPIO Pin Data Input Register (SIU_GPDI133)	8	R	See section	8.2.73/407
E86	GPIO Pin Data Input Register (SIU_GPDI134)	8	R	See section	8.2.73/407
E87	GPIO Pin Data Input Register (SIU_GPDI135)	8	R	See section	8.2.73/407
E88	GPIO Pin Data Input Register (SIU_GPDI136)	8	R	See section	8.2.73/407
E89	GPIO Pin Data Input Register (SIU_GPDI137)	8	R	See section	8.2.73/407
E8A	GPIO Pin Data Input Register (SIU_GPDI138)	8	R	See section	8.2.73/407
E8B	GPIO Pin Data Input Register (SIU_GPDI139)	8	R	See section	8.2.73/407
E8C	GPIO Pin Data Input Register (SIU_GPDI140)	8	R	See section	8.2.73/407
E8D	GPIO Pin Data Input Register (SIU_GPDI141)	8	R	See section	8.2.73/407
E8E	GPIO Pin Data Input Register (SIU_GPDI142)	8	R	See section	8.2.73/407
E8F	GPIO Pin Data Input Register (SIU_GPDI143)	8	R	See section	8.2.73/407

Table continues on the next page...

SIU memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
E90	GPIO Pin Data Input Register (SIU_GPDI144)	8	R	See section	8.2.73/407
E91	GPIO Pin Data Input Register (SIU_GPDI145)	8	R	See section	8.2.73/407
E92	GPIO Pin Data Input Register (SIU_GPDI146)	8	R	See section	8.2.73/407
E93	GPIO Pin Data Input Register (SIU_GPDI147)	8	R	See section	8.2.73/407
E94	GPIO Pin Data Input Register (SIU_GPDI148)	8	R	See section	8.2.73/407
E95	GPIO Pin Data Input Register (SIU_GPDI149)	8	R	See section	8.2.73/407
E96	GPIO Pin Data Input Register (SIU_GPDI150)	8	R	See section	8.2.73/407
E97	GPIO Pin Data Input Register (SIU_GPDI151)	8	R	See section	8.2.73/407
E98	GPIO Pin Data Input Register (SIU_GPDI152)	8	R	See section	8.2.73/407
E99	GPIO Pin Data Input Register (SIU_GPDI153)	8	R	See section	8.2.73/407
E9A	GPIO Pin Data Input Register (SIU_GPDI154)	8	R	See section	8.2.73/407
E9B	GPIO Pin Data Input Register (SIU_GPDI155)	8	R	See section	8.2.73/407
E9C	GPIO Pin Data Input Register (SIU_GPDI156)	8	R	See section	8.2.73/407
E9D	GPIO Pin Data Input Register (SIU_GPDI157)	8	R	See section	8.2.73/407
E9E	GPIO Pin Data Input Register (SIU_GPDI158)	8	R	See section	8.2.73/407
E9F	GPIO Pin Data Input Register (SIU_GPDI159)	8	R	See section	8.2.73/407
EA0	GPIO Pin Data Input Register (SIU_GPDI160)	8	R	See section	8.2.73/407
EA1	GPIO Pin Data Input Register (SIU_GPDI161)	8	R	See section	8.2.73/407
EA2	GPIO Pin Data Input Register (SIU_GPDI162)	8	R	See section	8.2.73/407
EA3	GPIO Pin Data Input Register (SIU_GPDI163)	8	R	See section	8.2.73/407
EA4	GPIO Pin Data Input Register (SIU_GPDI164)	8	R	See section	8.2.73/407
EA5	GPIO Pin Data Input Register (SIU_GPDI165)	8	R	See section	8.2.73/407
EA6	GPIO Pin Data Input Register (SIU_GPDI166)	8	R	See section	8.2.73/407
EA7	GPIO Pin Data Input Register (SIU_GPDI167)	8	R	See section	8.2.73/407
EA8	GPIO Pin Data Input Register (SIU_GPDI168)	8	R	See section	8.2.73/407
EA9	GPIO Pin Data Input Register (SIU_GPDI169)	8	R	See section	8.2.73/407
EAA	GPIO Pin Data Input Register (SIU_GPDI170)	8	R	See section	8.2.73/407
EAB	GPIO Pin Data Input Register (SIU_GPDI171)	8	R	See section	8.2.73/407
EAC	GPIO Pin Data Input Register (SIU_GPDI172)	8	R	See section	8.2.73/407
EAD	GPIO Pin Data Input Register (SIU_GPDI173)	8	R	See section	8.2.73/407
EAE	GPIO Pin Data Input Register (SIU_GPDI174)	8	R	See section	8.2.73/407
EAF	GPIO Pin Data Input Register (SIU_GPDI175)	8	R	See section	8.2.73/407
EB0	GPIO Pin Data Input Register (SIU_GPDI176)	8	R	See section	8.2.73/407
EB1	GPIO Pin Data Input Register (SIU_GPDI177)	8	R	See section	8.2.73/407
EB2	GPIO Pin Data Input Register (SIU_GPDI178)	8	R	See section	8.2.73/407
EB3	GPIO Pin Data Input Register (SIU_GPDI179)	8	R	See section	8.2.73/407
EB4	GPIO Pin Data Input Register (SIU_GPDI180)	8	R	See section	8.2.73/407
EB5	GPIO Pin Data Input Register (SIU_GPDI181)	8	R	See section	8.2.73/407
EB6	GPIO Pin Data Input Register (SIU_GPDI182)	8	R	See section	8.2.73/407

Table continues on the next page...

SIU memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
EB7	GPIO Pin Data Input Register (SIU_GPDI183)	8	R	See section	8.2.73/407
EB8	GPIO Pin Data Input Register (SIU_GPDI184)	8	R	See section	8.2.73/407
EB9	GPIO Pin Data Input Register (SIU_GPDI185)	8	R	See section	8.2.73/407
EBA	GPIO Pin Data Input Register (SIU_GPDI186)	8	R	See section	8.2.73/407
EBB	GPIO Pin Data Input Register (SIU_GPDI187)	8	R	See section	8.2.73/407
EBC	GPIO Pin Data Input Register (SIU_GPDI188)	8	R	See section	8.2.73/407
EBD	GPIO Pin Data Input Register (SIU_GPDI189)	8	R	See section	8.2.73/407
EBE	GPIO Pin Data Input Register (SIU_GPDI190)	8	R	See section	8.2.73/407
EBF	GPIO Pin Data Input Register (SIU_GPDI191)	8	R	See section	8.2.73/407
EC0	GPIO Pin Data Input Register (SIU_GPDI192)	8	R	See section	8.2.73/407
EC1	GPIO Pin Data Input Register (SIU_GPDI193)	8	R	See section	8.2.73/407
EC2	GPIO Pin Data Input Register (SIU_GPDI194)	8	R	See section	8.2.73/407
EC3	GPIO Pin Data Input Register (SIU_GPDI195)	8	R	See section	8.2.73/407
EC4	GPIO Pin Data Input Register (SIU_GPDI196)	8	R	See section	8.2.73/407
EC5	GPIO Pin Data Input Register (SIU_GPDI197)	8	R	See section	8.2.73/407
EC6	GPIO Pin Data Input Register (SIU_GPDI198)	8	R	See section	8.2.73/407
EC7	GPIO Pin Data Input Register (SIU_GPDI199)	8	R	See section	8.2.73/407
EC8	GPIO Pin Data Input Register (SIU_GPDI200)	8	R	See section	8.2.73/407
EC9	GPIO Pin Data Input Register (SIU_GPDI201)	8	R	See section	8.2.73/407
ECA	GPIO Pin Data Input Register (SIU_GPDI202)	8	R	See section	8.2.73/407
ECB	GPIO Pin Data Input Register (SIU_GPDI203)	8	R	See section	8.2.73/407
ECC	GPIO Pin Data Input Register (SIU_GPDI204)	8	R	See section	8.2.73/407
ECD	GPIO Pin Data Input Register (SIU_GPDI205)	8	R	See section	8.2.73/407
ECE	GPIO Pin Data Input Register (SIU_GPDI206)	8	R	See section	8.2.73/407
ECF	GPIO Pin Data Input Register (SIU_GPDI207)	8	R	See section	8.2.73/407
ED0	GPIO Pin Data Input Register (SIU_GPDI208)	8	R	See section	8.2.73/407
ED1	GPIO Pin Data Input Register (SIU_GPDI209)	8	R	See section	8.2.73/407
ED2	GPIO Pin Data Input Register (SIU_GPDI210)	8	R	See section	8.2.73/407
ED3	GPIO Pin Data Input Register (SIU_GPDI211)	8	R	See section	8.2.73/407
ED4	GPIO Pin Data Input Register (SIU_GPDI212)	8	R	See section	8.2.73/407
ED5	GPIO Pin Data Input Register (SIU_GPDI213)	8	R	See section	8.2.73/407
ED6	GPIO Pin Data Input Register (SIU_GPDI214)	8	R	See section	8.2.73/407
ED7	GPIO Pin Data Input Register (SIU_GPDI215)	8	R	See section	8.2.73/407
ED8	GPIO Pin Data Input Register (SIU_GPDI216)	8	R	See section	8.2.73/407
ED9	GPIO Pin Data Input Register (SIU_GPDI217)	8	R	See section	8.2.73/407
EDA	GPIO Pin Data Input Register (SIU_GPDI218)	8	R	See section	8.2.73/407
EDB	GPIO Pin Data Input Register (SIU_GPDI219)	8	R	See section	8.2.73/407
EDC	GPIO Pin Data Input Register (SIU_GPDI220)	8	R	See section	8.2.73/407
EDD	GPIO Pin Data Input Register (SIU_GPDI221)	8	R	See section	8.2.73/407

Table continues on the next page...

SIU memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
EDE	GPIO Pin Data Input Register (SIU_GPDI222)	8	R	See section	8.2.73/407
EDF	GPIO Pin Data Input Register (SIU_GPDI223)	8	R	See section	8.2.73/407
EE0	GPIO Pin Data Input Register (SIU_GPDI224)	8	R	See section	8.2.73/407
EE1	GPIO Pin Data Input Register (SIU_GPDI225)	8	R	See section	8.2.73/407
EE2	GPIO Pin Data Input Register (SIU_GPDI226)	8	R	See section	8.2.73/407
EE3	GPIO Pin Data Input Register (SIU_GPDI227)	8	R	See section	8.2.73/407
EE4	GPIO Pin Data Input Register (SIU_GPDI228)	8	R	See section	8.2.73/407
EE5	GPIO Pin Data Input Register (SIU_GPDI229)	8	R	See section	8.2.73/407
EE6	GPIO Pin Data Input Register (SIU_GPDI230)	8	R	See section	8.2.73/407
EE7	GPIO Pin Data Input Register (SIU_GPDI231)	8	R	See section	8.2.73/407
EE8	GPIO Pin Data Input Register (SIU_GPDI232)	8	R	See section	8.2.73/407
EE9	GPIO Pin Data Input Register (SIU_GPDI233)	8	R	See section	8.2.73/407
EEA	GPIO Pin Data Input Register (SIU_GPDI234)	8	R	See section	8.2.73/407
EEB	GPIO Pin Data Input Register (SIU_GPDI235)	8	R	See section	8.2.73/407
EEC	GPIO Pin Data Input Register (SIU_GPDI236)	8	R	See section	8.2.73/407
EED	GPIO Pin Data Input Register (SIU_GPDI237)	8	R	See section	8.2.73/407
EEE	GPIO Pin Data Input Register (SIU_GPDI238)	8	R	See section	8.2.73/407
EEF	GPIO Pin Data Input Register (SIU_GPDI239)	8	R	See section	8.2.73/407
EF0	GPIO Pin Data Input Register (SIU_GPDI240)	8	R	See section	8.2.73/407
EF1	GPIO Pin Data Input Register (SIU_GPDI241)	8	R	See section	8.2.73/407
EF2	GPIO Pin Data Input Register (SIU_GPDI242)	8	R	See section	8.2.73/407
EF3	GPIO Pin Data Input Register (SIU_GPDI243)	8	R	See section	8.2.73/407
EF4	GPIO Pin Data Input Register (SIU_GPDI244)	8	R	See section	8.2.73/407
EF5	GPIO Pin Data Input Register (SIU_GPDI245)	8	R	See section	8.2.73/407
EF6	GPIO Pin Data Input Register (SIU_GPDI246)	8	R	See section	8.2.73/407
EF7	GPIO Pin Data Input Register (SIU_GPDI247)	8	R	See section	8.2.73/407
EF8	GPIO Pin Data Input Register (SIU_GPDI248)	8	R	See section	8.2.73/407
EF9	GPIO Pin Data Input Register (SIU_GPDI249)	8	R	See section	8.2.73/407
EFA	GPIO Pin Data Input Register (SIU_GPDI250)	8	R	See section	8.2.73/407
EFB	GPIO Pin Data Input Register (SIU_GPDI251)	8	R	See section	8.2.73/407
EFC	GPIO Pin Data Input Register (SIU_GPDI252)	8	R	See section	8.2.73/407
EFD	GPIO Pin Data Input Register (SIU_GPDI253)	8	R	See section	8.2.73/407
EFE	GPIO Pin Data Input Register (SIU_GPDI254)	8	R	See section	8.2.73/407
EFF	GPIO Pin Data Input Register (SIU_GPDI255)	8	R	See section	8.2.73/407
F00	GPIO Pin Data Input Register (SIU_GPDI256)	8	R	See section	8.2.73/407
F01	GPIO Pin Data Input Register (SIU_GPDI257)	8	R	See section	8.2.73/407
F02	GPIO Pin Data Input Register (SIU_GPDI258)	8	R	See section	8.2.73/407
F03	GPIO Pin Data Input Register (SIU_GPDI259)	8	R	See section	8.2.73/407
F04	GPIO Pin Data Input Register (SIU_GPDI260)	8	R	See section	8.2.73/407

Table continues on the next page...

SIU memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
F05	GPIO Pin Data Input Register (SIU_GPDI261)	8	R	See section	8.2.73/407
F06	GPIO Pin Data Input Register (SIU_GPDI262)	8	R	See section	8.2.73/407
F07	GPIO Pin Data Input Register (SIU_GPDI263)	8	R	See section	8.2.73/407
F08	GPIO Pin Data Input Register (SIU_GPDI264)	8	R	See section	8.2.73/407
F09	GPIO Pin Data Input Register (SIU_GPDI265)	8	R	See section	8.2.73/407
F0A	GPIO Pin Data Input Register (SIU_GPDI266)	8	R	See section	8.2.73/407
F0B	GPIO Pin Data Input Register (SIU_GPDI267)	8	R	See section	8.2.73/407
F0C	GPIO Pin Data Input Register (SIU_GPDI268)	8	R	See section	8.2.73/407
F0D	GPIO Pin Data Input Register (SIU_GPDI269)	8	R	See section	8.2.73/407
F0E	GPIO Pin Data Input Register (SIU_GPDI270)	8	R	See section	8.2.73/407
F0F	GPIO Pin Data Input Register (SIU_GPDI271)	8	R	See section	8.2.73/407
F10	GPIO Pin Data Input Register (SIU_GPDI272)	8	R	See section	8.2.73/407
F11	GPIO Pin Data Input Register (SIU_GPDI273)	8	R	See section	8.2.73/407
F12	GPIO Pin Data Input Register (SIU_GPDI274)	8	R	See section	8.2.73/407
F13	GPIO Pin Data Input Register (SIU_GPDI275)	8	R	See section	8.2.73/407
F14	GPIO Pin Data Input Register (SIU_GPDI276)	8	R	See section	8.2.73/407
F15	GPIO Pin Data Input Register (SIU_GPDI277)	8	R	See section	8.2.73/407
F16	GPIO Pin Data Input Register (SIU_GPDI278)	8	R	See section	8.2.73/407
F17	GPIO Pin Data Input Register (SIU_GPDI279)	8	R	See section	8.2.73/407
F18	GPIO Pin Data Input Register (SIU_GPDI280)	8	R	See section	8.2.73/407
F19	GPIO Pin Data Input Register (SIU_GPDI281)	8	R	See section	8.2.73/407
F1A	GPIO Pin Data Input Register (SIU_GPDI282)	8	R	See section	8.2.73/407
F1B	GPIO Pin Data Input Register (SIU_GPDI283)	8	R	See section	8.2.73/407
F1C	GPIO Pin Data Input Register (SIU_GPDI284)	8	R	See section	8.2.73/407
F1D	GPIO Pin Data Input Register (SIU_GPDI285)	8	R	See section	8.2.73/407
F1E	GPIO Pin Data Input Register (SIU_GPDI286)	8	R	See section	8.2.73/407
F1F	GPIO Pin Data Input Register (SIU_GPDI287)	8	R	See section	8.2.73/407
F20	GPIO Pin Data Input Register (SIU_GPDI288)	8	R	See section	8.2.73/407
F21	GPIO Pin Data Input Register (SIU_GPDI289)	8	R	See section	8.2.73/407
F22	GPIO Pin Data Input Register (SIU_GPDI290)	8	R	See section	8.2.73/407
F23	GPIO Pin Data Input Register (SIU_GPDI291)	8	R	See section	8.2.73/407
F24	GPIO Pin Data Input Register (SIU_GPDI292)	8	R	See section	8.2.73/407
F25	GPIO Pin Data Input Register (SIU_GPDI293)	8	R	See section	8.2.73/407
F26	GPIO Pin Data Input Register (SIU_GPDI294)	8	R	See section	8.2.73/407
F27	GPIO Pin Data Input Register (SIU_GPDI295)	8	R	See section	8.2.73/407
F28	GPIO Pin Data Input Register (SIU_GPDI296)	8	R	See section	8.2.73/407
F29	GPIO Pin Data Input Register (SIU_GPDI297)	8	R	See section	8.2.73/407
F2A	GPIO Pin Data Input Register (SIU_GPDI298)	8	R	See section	8.2.73/407
F2B	GPIO Pin Data Input Register (SIU_GPDI299)	8	R	See section	8.2.73/407

Table continues on the next page...

SIU memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
F2C	GPIO Pin Data Input Register (SIU_GPDI300)	8	R	See section	8.2.73/407
F2D	GPIO Pin Data Input Register (SIU_GPDI301)	8	R	See section	8.2.73/407
F2E	GPIO Pin Data Input Register (SIU_GPDI302)	8	R	See section	8.2.73/407
F2F	GPIO Pin Data Input Register (SIU_GPDI303)	8	R	See section	8.2.73/407
F30	GPIO Pin Data Input Register (SIU_GPDI304)	8	R	See section	8.2.73/407
F31	GPIO Pin Data Input Register (SIU_GPDI305)	8	R	See section	8.2.73/407
F32	GPIO Pin Data Input Register (SIU_GPDI306)	8	R	See section	8.2.73/407
F33	GPIO Pin Data Input Register (SIU_GPDI307)	8	R	See section	8.2.73/407
F34	GPIO Pin Data Input Register (SIU_GPDI308)	8	R	See section	8.2.73/407
F35	GPIO Pin Data Input Register (SIU_GPDI309)	8	R	See section	8.2.73/407
F36	GPIO Pin Data Input Register (SIU_GPDI310)	8	R	See section	8.2.73/407
F37	GPIO Pin Data Input Register (SIU_GPDI311)	8	R	See section	8.2.73/407
F38	GPIO Pin Data Input Register (SIU_GPDI312)	8	R	See section	8.2.73/407
F39	GPIO Pin Data Input Register (SIU_GPDI313)	8	R	See section	8.2.73/407
F3A	GPIO Pin Data Input Register (SIU_GPDI314)	8	R	See section	8.2.73/407
F3B	GPIO Pin Data Input Register (SIU_GPDI315)	8	R	See section	8.2.73/407
F3C	GPIO Pin Data Input Register (SIU_GPDI316)	8	R	See section	8.2.73/407
F3D	GPIO Pin Data Input Register (SIU_GPDI317)	8	R	See section	8.2.73/407
F3E	GPIO Pin Data Input Register (SIU_GPDI318)	8	R	See section	8.2.73/407
F3F	GPIO Pin Data Input Register (SIU_GPDI319)	8	R	See section	8.2.73/407
F40	GPIO Pin Data Input Register (SIU_GPDI320)	8	R	See section	8.2.73/407
F41	GPIO Pin Data Input Register (SIU_GPDI321)	8	R	See section	8.2.73/407
F42	GPIO Pin Data Input Register (SIU_GPDI322)	8	R	See section	8.2.73/407
F43	GPIO Pin Data Input Register (SIU_GPDI323)	8	R	See section	8.2.73/407
F44	GPIO Pin Data Input Register (SIU_GPDI324)	8	R	See section	8.2.73/407
F45	GPIO Pin Data Input Register (SIU_GPDI325)	8	R	See section	8.2.73/407
F46	GPIO Pin Data Input Register (SIU_GPDI326)	8	R	See section	8.2.73/407
F47	GPIO Pin Data Input Register (SIU_GPDI327)	8	R	See section	8.2.73/407
F48	GPIO Pin Data Input Register (SIU_GPDI328)	8	R	See section	8.2.73/407
F49	GPIO Pin Data Input Register (SIU_GPDI329)	8	R	See section	8.2.73/407
F4A	GPIO Pin Data Input Register (SIU_GPDI330)	8	R	See section	8.2.73/407
F4B	GPIO Pin Data Input Register (SIU_GPDI331)	8	R	See section	8.2.73/407
F4C	GPIO Pin Data Input Register (SIU_GPDI332)	8	R	See section	8.2.73/407
F4D	GPIO Pin Data Input Register (SIU_GPDI333)	8	R	See section	8.2.73/407
F4E	GPIO Pin Data Input Register (SIU_GPDI334)	8	R	See section	8.2.73/407
F4F	GPIO Pin Data Input Register (SIU_GPDI335)	8	R	See section	8.2.73/407
F50	GPIO Pin Data Input Register (SIU_GPDI336)	8	R	See section	8.2.73/407
F51	GPIO Pin Data Input Register (SIU_GPDI337)	8	R	See section	8.2.73/407
F52	GPIO Pin Data Input Register (SIU_GPDI338)	8	R	See section	8.2.73/407

Table continues on the next page...

SIU memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
F53	GPIO Pin Data Input Register (SIU_GPDI339)	8	R	See section	8.2.73/407
F54	GPIO Pin Data Input Register (SIU_GPDI340)	8	R	See section	8.2.73/407
F55	GPIO Pin Data Input Register (SIU_GPDI341)	8	R	See section	8.2.73/407
F56	GPIO Pin Data Input Register (SIU_GPDI342)	8	R	See section	8.2.73/407
F57	GPIO Pin Data Input Register (SIU_GPDI343)	8	R	See section	8.2.73/407
F58	GPIO Pin Data Input Register (SIU_GPDI344)	8	R	See section	8.2.73/407
F59	GPIO Pin Data Input Register (SIU_GPDI345)	8	R	See section	8.2.73/407
F5A	GPIO Pin Data Input Register (SIU_GPDI346)	8	R	See section	8.2.73/407
F5B	GPIO Pin Data Input Register (SIU_GPDI347)	8	R	See section	8.2.73/407
F5C	GPIO Pin Data Input Register (SIU_GPDI348)	8	R	See section	8.2.73/407
F5D	GPIO Pin Data Input Register (SIU_GPDI349)	8	R	See section	8.2.73/407
F5E	GPIO Pin Data Input Register (SIU_GPDI350)	8	R	See section	8.2.73/407
F5F	GPIO Pin Data Input Register (SIU_GPDI351)	8	R	See section	8.2.73/407
F60	GPIO Pin Data Input Register (SIU_GPDI352)	8	R	See section	8.2.73/407
F61	GPIO Pin Data Input Register (SIU_GPDI353)	8	R	See section	8.2.73/407
F62	GPIO Pin Data Input Register (SIU_GPDI354)	8	R	See section	8.2.73/407
F63	GPIO Pin Data Input Register (SIU_GPDI355)	8	R	See section	8.2.73/407
F64	GPIO Pin Data Input Register (SIU_GPDI356)	8	R	See section	8.2.73/407
F65	GPIO Pin Data Input Register (SIU_GPDI357)	8	R	See section	8.2.73/407
F66	GPIO Pin Data Input Register (SIU_GPDI358)	8	R	See section	8.2.73/407
F67	GPIO Pin Data Input Register (SIU_GPDI359)	8	R	See section	8.2.73/407
F68	GPIO Pin Data Input Register (SIU_GPDI360)	8	R	See section	8.2.73/407
F69	GPIO Pin Data Input Register (SIU_GPDI361)	8	R	See section	8.2.73/407
F6A	GPIO Pin Data Input Register (SIU_GPDI362)	8	R	See section	8.2.73/407
F6B	GPIO Pin Data Input Register (SIU_GPDI363)	8	R	See section	8.2.73/407
F6C	GPIO Pin Data Input Register (SIU_GPDI364)	8	R	See section	8.2.73/407
F6D	GPIO Pin Data Input Register (SIU_GPDI365)	8	R	See section	8.2.73/407
F6E	GPIO Pin Data Input Register (SIU_GPDI366)	8	R	See section	8.2.73/407
F6F	GPIO Pin Data Input Register (SIU_GPDI367)	8	R	See section	8.2.73/407
F70	GPIO Pin Data Input Register (SIU_GPDI368)	8	R	See section	8.2.73/407
F71	GPIO Pin Data Input Register (SIU_GPDI369)	8	R	See section	8.2.73/407
F72	GPIO Pin Data Input Register (SIU_GPDI370)	8	R	See section	8.2.73/407
F73	GPIO Pin Data Input Register (SIU_GPDI371)	8	R	See section	8.2.73/407
F74	GPIO Pin Data Input Register (SIU_GPDI372)	8	R	See section	8.2.73/407
F75	GPIO Pin Data Input Register (SIU_GPDI373)	8	R	See section	8.2.73/407
F76	GPIO Pin Data Input Register (SIU_GPDI374)	8	R	See section	8.2.73/407
F77	GPIO Pin Data Input Register (SIU_GPDI375)	8	R	See section	8.2.73/407
F78	GPIO Pin Data Input Register (SIU_GPDI376)	8	R	See section	8.2.73/407
F79	GPIO Pin Data Input Register (SIU_GPDI377)	8	R	See section	8.2.73/407

Table continues on the next page...

SIU memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
F7A	GPIO Pin Data Input Register (SIU_GPDI378)	8	R	See section	8.2.73/407
F7B	GPIO Pin Data Input Register (SIU_GPDI379)	8	R	See section	8.2.73/407
F7C	GPIO Pin Data Input Register (SIU_GPDI380)	8	R	See section	8.2.73/407
F7D	GPIO Pin Data Input Register (SIU_GPDI381)	8	R	See section	8.2.73/407
F7E	GPIO Pin Data Input Register (SIU_GPDI382)	8	R	See section	8.2.73/407
F7F	GPIO Pin Data Input Register (SIU_GPDI383)	8	R	See section	8.2.73/407
F80	GPIO Pin Data Input Register (SIU_GPDI384)	8	R	See section	8.2.73/407
F81	GPIO Pin Data Input Register (SIU_GPDI385)	8	R	See section	8.2.73/407
F82	GPIO Pin Data Input Register (SIU_GPDI386)	8	R	See section	8.2.73/407
F83	GPIO Pin Data Input Register (SIU_GPDI387)	8	R	See section	8.2.73/407
F84	GPIO Pin Data Input Register (SIU_GPDI388)	8	R	See section	8.2.73/407
F85	GPIO Pin Data Input Register (SIU_GPDI389)	8	R	See section	8.2.73/407
F86	GPIO Pin Data Input Register (SIU_GPDI390)	8	R	See section	8.2.73/407
F87	GPIO Pin Data Input Register (SIU_GPDI391)	8	R	See section	8.2.73/407
F88	GPIO Pin Data Input Register (SIU_GPDI392)	8	R	See section	8.2.73/407
F89	GPIO Pin Data Input Register (SIU_GPDI393)	8	R	See section	8.2.73/407
F8A	GPIO Pin Data Input Register (SIU_GPDI394)	8	R	See section	8.2.73/407
F8B	GPIO Pin Data Input Register (SIU_GPDI395)	8	R	See section	8.2.73/407
F8C	GPIO Pin Data Input Register (SIU_GPDI396)	8	R	See section	8.2.73/407
F8D	GPIO Pin Data Input Register (SIU_GPDI397)	8	R	See section	8.2.73/407
F8E	GPIO Pin Data Input Register (SIU_GPDI398)	8	R	See section	8.2.73/407
F8F	GPIO Pin Data Input Register (SIU_GPDI399)	8	R	See section	8.2.73/407
F90	GPIO Pin Data Input Register (SIU_GPDI400)	8	R	See section	8.2.73/407
F91	GPIO Pin Data Input Register (SIU_GPDI401)	8	R	See section	8.2.73/407
F92	GPIO Pin Data Input Register (SIU_GPDI402)	8	R	See section	8.2.73/407
F93	GPIO Pin Data Input Register (SIU_GPDI403)	8	R	See section	8.2.73/407
F94	GPIO Pin Data Input Register (SIU_GPDI404)	8	R	See section	8.2.73/407
F95	GPIO Pin Data Input Register (SIU_GPDI405)	8	R	See section	8.2.73/407
F96	GPIO Pin Data Input Register (SIU_GPDI406)	8	R	See section	8.2.73/407
F97	GPIO Pin Data Input Register (SIU_GPDI407)	8	R	See section	8.2.73/407
F98	GPIO Pin Data Input Register (SIU_GPDI408)	8	R	See section	8.2.73/407
F99	GPIO Pin Data Input Register (SIU_GPDI409)	8	R	See section	8.2.73/407
F9A	GPIO Pin Data Input Register (SIU_GPDI410)	8	R	See section	8.2.73/407
F9B	GPIO Pin Data Input Register (SIU_GPDI411)	8	R	See section	8.2.73/407
F9C	GPIO Pin Data Input Register (SIU_GPDI412)	8	R	See section	8.2.73/407
F9D	GPIO Pin Data Input Register (SIU_GPDI413)	8	R	See section	8.2.73/407
F9E	GPIO Pin Data Input Register (SIU_GPDI414)	8	R	See section	8.2.73/407
F9F	GPIO Pin Data Input Register (SIU_GPDI415)	8	R	See section	8.2.73/407
FA0	GPIO Pin Data Input Register (SIU_GPDI416)	8	R	See section	8.2.73/407

Table continues on the next page...

SIU memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
FA1	GPIO Pin Data Input Register (SIU_GPDI417)	8	R	See section	8.2.73/407
FA2	GPIO Pin Data Input Register (SIU_GPDI418)	8	R	See section	8.2.73/407
FA3	GPIO Pin Data Input Register (SIU_GPDI419)	8	R	See section	8.2.73/407
FA4	GPIO Pin Data Input Register (SIU_GPDI420)	8	R	See section	8.2.73/407
FA5	GPIO Pin Data Input Register (SIU_GPDI421)	8	R	See section	8.2.73/407
FA6	GPIO Pin Data Input Register (SIU_GPDI422)	8	R	See section	8.2.73/407
FA7	GPIO Pin Data Input Register (SIU_GPDI423)	8	R	See section	8.2.73/407
FA8	GPIO Pin Data Input Register (SIU_GPDI424)	8	R	See section	8.2.73/407
FA9	GPIO Pin Data Input Register (SIU_GPDI425)	8	R	See section	8.2.73/407
FAA	GPIO Pin Data Input Register (SIU_GPDI426)	8	R	See section	8.2.73/407
FAB	GPIO Pin Data Input Register (SIU_GPDI427)	8	R	See section	8.2.73/407
FAC	GPIO Pin Data Input Register (SIU_GPDI428)	8	R	See section	8.2.73/407
FAD	GPIO Pin Data Input Register (SIU_GPDI429)	8	R	See section	8.2.73/407
FAE	GPIO Pin Data Input Register (SIU_GPDI430)	8	R	See section	8.2.73/407
FAF	GPIO Pin Data Input Register (SIU_GPDI431)	8	R	See section	8.2.73/407
FB0	GPIO Pin Data Input Register (SIU_GPDI432)	8	R	See section	8.2.73/407
FB1	GPIO Pin Data Input Register (SIU_GPDI433)	8	R	See section	8.2.73/407
FB2	GPIO Pin Data Input Register (SIU_GPDI434)	8	R	See section	8.2.73/407
FB3	GPIO Pin Data Input Register (SIU_GPDI435)	8	R	See section	8.2.73/407
FB4	GPIO Pin Data Input Register (SIU_GPDI436)	8	R	See section	8.2.73/407
FB5	GPIO Pin Data Input Register (SIU_GPDI437)	8	R	See section	8.2.73/407
FB6	GPIO Pin Data Input Register (SIU_GPDI438)	8	R	See section	8.2.73/407
FB7	GPIO Pin Data Input Register (SIU_GPDI439)	8	R	See section	8.2.73/407
FB8	GPIO Pin Data Input Register (SIU_GPDI440)	8	R	See section	8.2.73/407
FB9	GPIO Pin Data Input Register (SIU_GPDI441)	8	R	See section	8.2.73/407
FBA	GPIO Pin Data Input Register (SIU_GPDI442)	8	R	See section	8.2.73/407
FBB	GPIO Pin Data Input Register (SIU_GPDI443)	8	R	See section	8.2.73/407
FBC	GPIO Pin Data Input Register (SIU_GPDI444)	8	R	See section	8.2.73/407
FBD	GPIO Pin Data Input Register (SIU_GPDI445)	8	R	See section	8.2.73/407
FBE	GPIO Pin Data Input Register (SIU_GPDI446)	8	R	See section	8.2.73/407
FBF	GPIO Pin Data Input Register (SIU_GPDI447)	8	R	See section	8.2.73/407
FC0	GPIO Pin Data Input Register (SIU_GPDI448)	8	R	See section	8.2.73/407
FC1	GPIO Pin Data Input Register (SIU_GPDI449)	8	R	See section	8.2.73/407
FC2	GPIO Pin Data Input Register (SIU_GPDI450)	8	R	See section	8.2.73/407
FC3	GPIO Pin Data Input Register (SIU_GPDI451)	8	R	See section	8.2.73/407
FC4	GPIO Pin Data Input Register (SIU_GPDI452)	8	R	See section	8.2.73/407
FC5	GPIO Pin Data Input Register (SIU_GPDI453)	8	R	See section	8.2.73/407
FC6	GPIO Pin Data Input Register (SIU_GPDI454)	8	R	See section	8.2.73/407
FC7	GPIO Pin Data Input Register (SIU_GPDI455)	8	R	See section	8.2.73/407

Table continues on the next page...

SIU memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
FC8	GPIO Pin Data Input Register (SIU_GPDI456)	8	R	See section	8.2.73/407
FC9	GPIO Pin Data Input Register (SIU_GPDI457)	8	R	See section	8.2.73/407
FCA	GPIO Pin Data Input Register (SIU_GPDI458)	8	R	See section	8.2.73/407
FCB	GPIO Pin Data Input Register (SIU_GPDI459)	8	R	See section	8.2.73/407
FCC	GPIO Pin Data Input Register (SIU_GPDI460)	8	R	See section	8.2.73/407
FCD	GPIO Pin Data Input Register (SIU_GPDI461)	8	R	See section	8.2.73/407
FCE	GPIO Pin Data Input Register (SIU_GPDI462)	8	R	See section	8.2.73/407
FCF	GPIO Pin Data Input Register (SIU_GPDI463)	8	R	See section	8.2.73/407
FD0	GPIO Pin Data Input Register (SIU_GPDI464)	8	R	See section	8.2.73/407
FD1	GPIO Pin Data Input Register (SIU_GPDI465)	8	R	See section	8.2.73/407
FD2	GPIO Pin Data Input Register (SIU_GPDI466)	8	R	See section	8.2.73/407
FD3	GPIO Pin Data Input Register (SIU_GPDI467)	8	R	See section	8.2.73/407
FD4	GPIO Pin Data Input Register (SIU_GPDI468)	8	R	See section	8.2.73/407
FD5	GPIO Pin Data Input Register (SIU_GPDI469)	8	R	See section	8.2.73/407
FD6	GPIO Pin Data Input Register (SIU_GPDI470)	8	R	See section	8.2.73/407
FD7	GPIO Pin Data Input Register (SIU_GPDI471)	8	R	See section	8.2.73/407
FD8	GPIO Pin Data Input Register (SIU_GPDI472)	8	R	See section	8.2.73/407
FD9	GPIO Pin Data Input Register (SIU_GPDI473)	8	R	See section	8.2.73/407
FDA	GPIO Pin Data Input Register (SIU_GPDI474)	8	R	See section	8.2.73/407
FDB	GPIO Pin Data Input Register (SIU_GPDI475)	8	R	See section	8.2.73/407
FDC	GPIO Pin Data Input Register (SIU_GPDI476)	8	R	See section	8.2.73/407
FDD	GPIO Pin Data Input Register (SIU_GPDI477)	8	R	See section	8.2.73/407
FDE	GPIO Pin Data Input Register (SIU_GPDI478)	8	R	See section	8.2.73/407
FDF	GPIO Pin Data Input Register (SIU_GPDI479)	8	R	See section	8.2.73/407
FE0	GPIO Pin Data Input Register (SIU_GPDI480)	8	R	See section	8.2.73/407
FE1	GPIO Pin Data Input Register (SIU_GPDI481)	8	R	See section	8.2.73/407
FE2	GPIO Pin Data Input Register (SIU_GPDI482)	8	R	See section	8.2.73/407
FE3	GPIO Pin Data Input Register (SIU_GPDI483)	8	R	See section	8.2.73/407
FE4	GPIO Pin Data Input Register (SIU_GPDI484)	8	R	See section	8.2.73/407
FE5	GPIO Pin Data Input Register (SIU_GPDI485)	8	R	See section	8.2.73/407
FE6	GPIO Pin Data Input Register (SIU_GPDI486)	8	R	See section	8.2.73/407
FE7	GPIO Pin Data Input Register (SIU_GPDI487)	8	R	See section	8.2.73/407
FE8	GPIO Pin Data Input Register (SIU_GPDI488)	8	R	See section	8.2.73/407
FE9	GPIO Pin Data Input Register (SIU_GPDI489)	8	R	See section	8.2.73/407
FEA	GPIO Pin Data Input Register (SIU_GPDI490)	8	R	See section	8.2.73/407
FEB	GPIO Pin Data Input Register (SIU_GPDI491)	8	R	See section	8.2.73/407
1200	Input Multiplexing Register0 (SIU_IMUX0)	32	R/W	0000_0000h	8.2.74/408
1204	Input Multiplexing Register1 (SIU_IMUX1)	32	R/W	0000_0000h	8.2.75/410
1208	Input Multiplexing Register2 (SIU_IMUX2)	32	R/W	0000_0000h	8.2.76/411

Table continues on the next page...

SIU memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
120C	Input Multiplexing Register3 (SIU_IMUX3)	32	R/W	0000_0000h	8.2.77/413
1210	Input Multiplexing Register4 (SIU_IMUX4)	32	R/W	0000_0000h	8.2.78/414
1214	Input Multiplexing Register5 (SIU_IMUX5)	32	R/W	0000_0000h	8.2.79/416
121C	Input Multiplexing Register7 (SIU_IMUX7)	32	R/W	0000_0000h	8.2.80/418
1228	Input Multiplexing Register10 (SIU_IMUX10)	32	R/W	0000_0000h	8.2.81/420
1230	Input Multiplexing Register12 (SIU_IMUX12)	32	R/W	0000_0000h	8.2.82/422

8.2.1 MCU Identification Register (SIU_MIDR)

The SIU_MIDR contains the part number and mask number revisions specific to the device. The part number is a read-only field that is mask programmed with the part number of the device on which the SIU resides. It is intended to help factory test and user code which are sensitive to part changes. The part number is changed if a new module is added to the device or a memory size is changed, for example. It is not changed for bug fixes or process changes.

The mask number is a read-only field that is mask programmed with the specific mask revision level of the device. It is intended to help factory test and user code which is sensitive to mask changes.

Address: 0h base + 4h offset = 4h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	PARTNUM																PKG			0			MASKNUM_ MAJOR				MASKNUM_ MINOR					
W																																
Reset	0	1	0	1	0	1	1	1	0	1	1	1	0	1	1	1	*	*	*	*	0	0	0	0	*	*	*	*	*	*	*	*

* Notes:

- MASKNUM_MINOR field: 1N45H = 0001b, 2N45H = 0010b, 3N45H = 0000b
- MASKNUM_MAJOR field: 1N45H = 0000b, 2N45H = 0000b, 3N45H = 0001b
- PKG field: 416 = 0110b, 516 = 1110b, 552 = 1100b

SIU_MIDR field descriptions

Field	Description
0–15 PARTNUM	MCU Part Number. Read-only, mask programmed part number of the MCU. MPC5777 reads 0x5777.
16–19 PKG	Package settings. Read-only, package information. 0110: 416-pin package. 1110: 516-pin package.

Table continues on the next page...

SIU_MIDR field descriptions (continued)

Field	Description
	1100: 552-pin package. All other settings reserved.
20–23 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
24–27 MASKNUM_ MAJOR	Major revision of MCU Mask. Read-only, mask programmed mask number of the MCU. Reads 0x0 for the initial mask set of the device, and changes sequentially for each mask set.
28–31 MASKNUM_ MINOR	Minor revision MCU Mask. Read-only, mask programmed mask number of the MCU. Reads 0x0 for the initial mask set of the device, and changes sequentially for each mask set.

8.2.2 Reset Status Register (SIU_RSR)

The Reset Status Register contains the sources of the most recent reset as well as a number of configuration bits defined by the Miscellaneous DCF record, which the SSCM reads from flash memory during reset. Alternatively, if the LEG bit of the Miscellaneous DCF record is set to 1, then the reset controller assumes that a backwards compatible legacy mode must be used. In this case, the values of the configuration bits are latched from package pins at the end of the reset cycle instead of coming from the DCF record.

The reset values of status bits are applied at POR. The reset value of the configuration bits (WKPCFG, XOSCHIGH, ABR, BOOTCFG) are loaded from the Miscellaneous DCF record or from external pins.

Address: 0h base + Ch offset = Ch

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	PORS	ERS				0				CPURS	0	FOSURS	FCCURS	STCURS	SSRS	SERF
W																w1c
Reset	1*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*

Memory Map and Register Definition

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	
R	WKPCFG								XOSC	0			XOSCHIGH	ABR	BOOTCFG		RGF
W																	w1c
Reset	*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	*	*	*	*	0*	

* Notes:

- The reset values of status bits are applied at POR.
- BOOTCFG field: The reset value of this configuration field is loaded from the Miscellaneous DCF record or from external pins.
- ABR field: The reset value of this configuration field is loaded from the Miscellaneous DCF record or from external pins.
- XOSCHIGH field: The reset value of this configuration field is loaded from the Miscellaneous DCF record or from external pins.
- WKPCFG field: The reset value of this configuration field is loaded from the Miscellaneous DCF record or from external pins.

SIU_RSR field descriptions

Field	Description
0 PORS	Power-on reset status. 0 Another reset source has been acknowledged by the reset controller since the last assertion of the power-on reset input. 1 The power-on reset input to the reset controller was asserted and no other reset source was acknowledged since that assertion of the power-on reset input, except an external reset.
1 ERS	External reset status. 0 The last reset source acknowledged by the reset controller was not a valid assertion of the RESET pin. 1 The last reset source acknowledged by the reset controller was a valid assertion of the RESET pin.
2–8 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
9 CPURS	Simultaneous Core 0 and Core 1 reset status. 0 The last reset source acknowledged by the reset controller was not a simultaneous reset attempt of both cores. 1 The last reset source acknowledged by the reset controller was a simultaneous reset attempt of both cores.
10 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

Table continues on the next page...

SIU_RSR field descriptions (continued)

Field	Description
11 FOSURS	FCCU Output Supervision Unit (FOSU) reset status. 0 The last reset source acknowledged by the reset controller was not a reset from the FOSU. 1 The last reset source acknowledged by the reset controller was a reset from the FOSU.
12 FCCURS	Failure Collection and Control Unit (FCCU) reset status. 0 The last reset source acknowledged by the reset controller was not a reset from the FCCU. 1 The last reset source acknowledged by the reset controller was a reset from the FCCU.
13 STCURS	Self Test Control Unit (STCU) reset status. 0 The last reset source acknowledged by the reset controller was not a reset from the STCU. 1 The last reset source acknowledged by the reset controller was a reset from the STCU.
14 SSRS	Software system reset status. 0 The last reset source acknowledged by the reset controller was not a software system reset. 1 The last reset source acknowledged by the reset controller was a software system reset.
15 SERF	Software external reset flag. 0 The software external reset input to the reset controller was not asserted, or this bit has been cleared by writing a 1 to it. 1 The software external reset input to the reset controller was asserted while this bit was zero.
16 WKPCFG	Weak pull-up configuration. The value of this bit is loaded from bit 12 of the Miscellaneous DCF record by the SSCM during reset. If the MSB bit of the DCF record (Legacy Bit) is set to '1', then the value of this bit is latched from the WKPCFG pin upon negation of reset, instead of being loaded from the DCF record. 0 Weak pulldown is the default setting. 1 Weak pullup is the default setting.
17–23 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
24 XOSC	External oscillator status. This is a set-once bit that is set after an internal counter (clocked by XOSC) finishes counting to 512 times the XOSC_CNT field of the Miscellaneous DCF record. When set, this flag indicates that the XOSC clock is stable and can be used as clock source for system and peripherals. 0 XOSC clock not stable yet. 1 XOSC clock is stable.
25–26 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
27 XOSCHIGH	XOSC at High Setting. The value of this bit is loaded from bit 21 of the Miscellaneous DCF record by the SSCM during reset. However, if the MSB of the DCF record (Legacy bit) is set to 1, then the value of this bit is latched from the PLLCFG[2] pin upon negation of reset instead of being loaded from the DCF record. NOTE: "Low" is the setting commonly used for crystals at 8 MHz, "medium" is commonly used for crystals greater than 8 MHz to 20 MHz, and "high" is commonly used for crystals greater than 20 MHz to 40 MHz. However, the user must characterize carefully to determine the best g_m setting for the intended application because crystal load capacitance, board layout, and other factors affect the g_m value that is needed. See the Data Sheet for additional information. 0 Crystal in the range of low to medium setting. 1 Crystal is at high setting.
28 ABR	Auto baud rate configuration. The value of this bit is loaded from bit 13 of the Miscellaneous DCF record by the SSCM during reset. If the MSB bit of the DCF record (Legacy Bit) is set to '1', then the value of this

Table continues on the next page...

SIU_RSR field descriptions (continued)

Field	Description
	bit is determined by the value of the EVTO pin upon negation of reset as follows. If no debug tool is attached to the device and EVTO is held at low logic level during reset, the ABR bit reset state is logic one. Otherwise it is zero. 0 Auto Baud Rate Disabled. 1 Auto Baud Rate Enabled.
29–30 BOOTCFG	Boot configuration. BOOTCFG[0:1] is used by the Boot Assist Module (BAM) to configure the boot mode. If the component life cycle is "in field", then the value of BOOTCFG is fixed to 0b00 (internal boot). For all other life cycles, the value of BOOTCFG is loaded from bits 14-15 of the Miscellaneous DCF record by the SSCM during reset. Alternatively, if the MSB bit of the Miscellaneous DCF record (Legacy Bit) is set to '1', then the value of this field is latched from the BOOTCFG pins upon negation of reset, instead of being loaded from the DCF record. 00 Internal boot mode. 01 Serial boot mode. 10 Non-muxed EBI boot mode (16-bit data, non-muxed address bus). 11 Muxed EBI boot mode (16-bit data, muxed address bus).
31 RGF	Reset glitch flag. Set by the reset controller when a glitch is detected on the RESET pin. This bit is cleared by the assertion of the power-on reset or a write of one to the RGF bit. 0 No glitch has been detected on the RESET pin. 1 A glitch has been detected on the RESET pin.

8.2.3 System Reset Control Register (SIU_SRCR)

The system reset control register configures whether a software system reset or a software external reset is generated. A software system reset uses an internal system reset. A software external reset asserts $\overline{\text{RSTOUT}}$.

Address: 0h base + 10h offset = 10h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0															
W	SSR	SER														
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R																
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

SIU_SRCR field descriptions

Field	Description
0 SSR	Software system reset. The software system reset is processed as a synchronous reset. Except for a software external reset, the bit automatically clears if any other reset source asserts. NOTE: The SSR bit always reads as zero. A write of zero to this bit has no effect. 0 No software system reset. 1 Generate a software internal system reset.
1 SER	Software external reset. Used to generate a Software External Reset. Writing 1 to this bit asserts RSTOUT for 2480 clock cycles, but the internal reset is not asserted. The bit is automatically cleared when the software external reset completes or any other reset source is asserted. After a software external reset has been initiated, RSTOUT negates if this bit is cleared before the 2480 clock cycles period expires. NOTE: Write 1 to the SER bit to generate a software external reset. A write of 0 to this bit has no effect. When reset completes, the SER bit is cleared to 0. 0 Do not generate a software external reset. 1 Generate a software external reset.
2–31 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

8.2.4 External IRQ Status Register (SIU_EISR)

The external interrupt status register is used to record edge-triggered events on the $\overline{\text{IRQ}}[0]-\overline{\text{IRQ}}[15]$ inputs to the SIU and record the critical interrupts NMI and SWT. When an edge-detect enable bit is set in the SIU_IREER or SIU_IFEER registers for an IRQ and an IRQ edge-event is detected, the flag bit is set in the SIU_EISR. The flag bits are cleared by writing 1 to the bit. Writing 0 has no effect.

The flag bit is set regardless of the state of the DMA or interrupt request enable bit in SIU_DIRER. The flag bit remains set until cleared by software or through the servicing of a DMA or interrupt request.

Address: 0h base + 14h offset = 14h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	NMI0	NMI1	0													
W	w1c	w1c														
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Memory Map and Register Definition

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	EIF15	EIF14	EIF13	EIF12	EIF11	EIF10	EIF9	EIF8	EIF7	EIF6	EIF5	EIF4	EIF3	EIF2	EIF1	EIF0
W	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

SIU_EISR field descriptions

Field	Description
0 NMI0	Core0 non-maskable interrupt flag. This bit is set when a NMI interrupt occurs on the NMI input pin of Core0. 0 No NMI event has occurred on the NMI input of Core0. 1 An NMI event has occurred on the NMI input of Core0.
1 NMI1	Core1 non-maskable interrupt flag. This bit is set when a NMI interrupt occurs on the NMI input pin of Core1. 0 No NMI event has occurred on the NMI input of Core1. 1 An NMI event has occurred on the NMI input of Core1.
2–15 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
16 EIF15	External interrupt request flag 15. This bit is set when an edge-triggered event occurs on the corresponding $\overline{\text{IRQ}}[15]$ input. Cleared by writing 1. 0 No edge triggered event has occurred on the corresponding $\overline{\text{IRQ}}$ pin. 1 An edge triggered event has occurred on the corresponding $\overline{\text{IRQ}}$ pin.
17 EIF14	External interrupt request flag 14. This bit is set when an edge-triggered event occurs on the corresponding $\overline{\text{IRQ}}[14]$ input. Cleared by writing 1. 0 No edge triggered event has occurred on the corresponding $\overline{\text{IRQ}}$ pin. 1 An edge triggered event has occurred on the corresponding $\overline{\text{IRQ}}$ pin.
18 EIF13	External interrupt request flag 13. This bit is set when an edge-triggered event occurs on the corresponding $\overline{\text{IRQ}}[13]$ input. Cleared by writing 1. 0 No edge triggered event has occurred on the corresponding $\overline{\text{IRQ}}$ pin. 1 An edge triggered event has occurred on the corresponding $\overline{\text{IRQ}}$ pin.
19 EIF12	External interrupt request flag 12. This bit is set when an edge-triggered event occurs on the corresponding $\overline{\text{IRQ}}[12]$ input. Cleared by writing 1. 0 No edge triggered event has occurred on the corresponding $\overline{\text{IRQ}}$ pin. 1 An edge triggered event has occurred on the corresponding $\overline{\text{IRQ}}$ pin.
20 EIF11	External interrupt request flag 11. This bit is set when an edge-triggered event occurs on the corresponding $\overline{\text{IRQ}}[11]$ input. Cleared by writing 1. 0 No edge triggered event has occurred on the corresponding $\overline{\text{IRQ}}$ pin. 1 An edge triggered event has occurred on the corresponding $\overline{\text{IRQ}}$ pin.
21 EIF10	External interrupt request flag 10. This bit is set when an edge-triggered event occurs on the corresponding $\overline{\text{IRQ}}[10]$ input. Cleared by writing 1.

Table continues on the next page...

SIU_EISR field descriptions (continued)

Field	Description
	0 No edge triggered event has occurred on the corresponding $\overline{\text{IRQ}}$ pin. 1 An edge triggered event has occurred on the corresponding $\overline{\text{IRQ}}$ pin.
22 EIF9	External interrupt request flag 9. This bit is set when an edge-triggered event occurs on the corresponding $\overline{\text{IRQ}}[9]$ input. Cleared by writing 1. 0 No edge triggered event has occurred on the corresponding $\overline{\text{IRQ}}$ pin. 1 An edge triggered event has occurred on the corresponding $\overline{\text{IRQ}}$ pin.
23 EIF8	External interrupt request flag 8. This bit is set when an edge-triggered event occurs on the corresponding $\overline{\text{IRQ}}[8]$ input. Cleared by writing 1. 0 No edge triggered event has occurred on the corresponding $\overline{\text{IRQ}}$ pin. 1 An edge triggered event has occurred on the corresponding $\overline{\text{IRQ}}$ pin.
24 EIF7	External interrupt request flag 7. This bit is set when an edge-triggered event occurs on the corresponding $\overline{\text{IRQ}}[7]$ input. Cleared by writing 1. 0 No edge triggered event has occurred on the corresponding $\overline{\text{IRQ}}$ pin. 1 An edge triggered event has occurred on the corresponding $\overline{\text{IRQ}}$ pin.
25 EIF6	External interrupt request flag 6. This bit is set when an edge-triggered event occurs on the corresponding $\overline{\text{IRQ}}[6]$ input. Cleared by writing 1. 0 No edge triggered event has occurred on the corresponding $\overline{\text{IRQ}}$ pin. 1 An edge triggered event has occurred on the corresponding $\overline{\text{IRQ}}$ pin.
26 EIF5	External interrupt request flag 5. This bit is set when an edge-triggered event occurs on the corresponding $\overline{\text{IRQ}}[5]$ input. Cleared by writing 1. 0 No edge triggered event has occurred on the corresponding $\overline{\text{IRQ}}$ pin. 1 An edge triggered event has occurred on the corresponding $\overline{\text{IRQ}}$ pin.
27 EIF4	External interrupt request flag 4. This bit is set when an edge-triggered event occurs on the corresponding $\overline{\text{IRQ}}[4]$ input. Cleared by writing 1. 0 No edge triggered event has occurred on the corresponding $\overline{\text{IRQ}}$ pin. 1 An edge triggered event has occurred on the corresponding $\overline{\text{IRQ}}$ pin.
28 EIF3	External interrupt request flag 3. This bit is set when an edge-triggered event occurs on the corresponding $\overline{\text{IRQ}}[3]$ input. Cleared by writing 1. 0 No edge triggered event has occurred on the corresponding $\overline{\text{IRQ}}$ pin. 1 An edge triggered event has occurred on the corresponding $\overline{\text{IRQ}}$ pin.
29 EIF2	External interrupt request flag 2. This bit is set when an edge-triggered event occurs on the corresponding $\overline{\text{IRQ}}[2]$ input. Cleared by writing 1. 0 No edge triggered event has occurred on the corresponding $\overline{\text{IRQ}}$ pin. 1 An edge triggered event has occurred on the corresponding $\overline{\text{IRQ}}$ pin.
30 EIF1	External interrupt request flag 1. This bit is set when an edge-triggered event occurs on the corresponding $\overline{\text{IRQ}}[1]$ input. Cleared by writing 1. 0 No edge triggered event has occurred on the corresponding $\overline{\text{IRQ}}$ pin. 1 An edge triggered event has occurred on the corresponding $\overline{\text{IRQ}}$ pin.
31 EIF0	External interrupt request flag 0. This bit is set when an edge-triggered event occurs on the corresponding $\overline{\text{IRQ}}[0]$ input. Cleared by writing 1.

Table continues on the next page...

SIU_EISR field descriptions (continued)

Field	Description
0	No edge triggered event has occurred on the corresponding IRQ pin.
1	An edge triggered event has occurred on the corresponding IRQ pin.

8.2.5 DMA/Interrupt Request Enable Register (SIU_DIRER)

The SIU_DIRER enables external signals on selected pins to generate DMA or interrupt requests, and the SWT modules to generate interrupt requests. The EIRE[0:3] bits enable requests to either the DMA or INTC modules, based on the value of the DIRSn bits in the SIU_DIRSR register. The EIRE[4:15] bits enable requests only to the INTC module. The NMISELn bits enable requests to either the critical or non-maskable interrupt inputs of the cores. For a pin to generate an interrupt, the appropriate bit in either SIU_IREER or SIU_IFEER registers must additionally be set.

Address: 0h base + 18h offset = 18h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	NMISEL8	NMISEL7	0						NMISEL0	NMISEL1	0					
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	EIRE15	EIRE14	EIRE13	EIRE12	EIRE11	EIRE10	EIRE9	EIRE8	EIRE7	EIRE6	EIRE5	EIRE4	EIRE3	EIRE2	EIRE1	EIRE0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

SIU_DIRER field descriptions

Field	Description
0 NMISEL8	Core0 non-maskable/critical interrupt selection (NMI comes from external pin). The NMI external pin can be routed to Core0 either as critical interrupt or as non-maskable interrupt (NMI). The NMISEL8 bit selects which of them is driven by the NMI pin. This bit is write-once. NOTE: NMISEL8 is a write-once bit. 0 Core0 NMI interrupt receives the signal from the NMI pin (IVOR1 core exception). 1 Core0 critical interrupt receives the signal from the NMI pin (IVOR0 core exception).
1 NMISEL7	Core1 non-maskable/critical interrupt selection (NMI comes from external pin). The NMI external pin can be routed to Core1 either as critical interrupt or as non-maskable interrupt (NMI). The NMISEL7 bit selects which of them is driven by the NMI pin. This bit is write-once.

Table continues on the next page...

SIU_DIRER field descriptions (continued)

Field	Description
	NOTE: NMISEL7 is a write-once bit. 0 Core1 NMI interrupt receives the signal from the NMI pin (IVOR1 core exception). 1 Core1 critical interrupt receives the signal from the NMI pin (IVOR0 core exception).
2–7 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
8 NMISEL0	Core0 non-maskable/critical interrupt selection (NMI comes from SWT_A). The Software Watchdog Timer A (SWT_A) interrupt output can be routed to Core0 either as critical interrupt or as non-maskable interrupt (NMI). The NMISEL0 bit selects which of them is driven by SWT_A. This bit is write-once. NOTE: NMISEL0 is a write-once bit. 0 Core0 NMI interrupt receives the signal from SWT_A (IVOR1 core exception). 1 Core0 critical interrupt receives the signal from SWT_A (IVOR0 core exception).
9 NMISEL1	Core1 non-maskable/critical interrupt selection (NMI comes from SWT_B). The Software Watchdog Timer B (SWT_B) interrupt output can be routed to Core1 either as critical interrupt or as non-maskable interrupt (NMI). The NMISEL1 bit selects which of them is driven by SWT_B. This bit is write-once. NOTE: NMISEL1 is a write-once bit. 0 Core1 NMI interrupt receives the signal from SWT_B (IVOR1 core exception). 1 Core1 critical interrupt receives the signal from SWT_B (IVOR0 core exception).
10–15 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
16 EIRE15	Interrupt request enable 15. This bit enables the assertion of an interrupt request when an edge-triggered event occurs on the $\overline{\text{IRQ}}[15]$ pin. 0 Interrupt request is disabled. 1 Interrupt request is enabled.
17 EIRE14	Interrupt request enable 14. This bit enables the assertion of an interrupt request when an edge-triggered event occurs on the $\overline{\text{IRQ}}[14]$ pin. 0 Interrupt request is disabled. 1 Interrupt request is enabled.
18 EIRE13	Interrupt request enable 13. This bit enables the assertion of an interrupt request when an edge-triggered event occurs on the $\overline{\text{IRQ}}[13]$ pin. 0 Interrupt request is disabled. 1 Interrupt request is enabled.
19 EIRE12	Interrupt request enable 12. This bit enables the assertion of an interrupt request when an edge-triggered event occurs on the $\overline{\text{IRQ}}[12]$ pin. 0 Interrupt request is disabled. 1 Interrupt request is enabled.
20 EIRE11	Interrupt request enable 11. This bit enables the assertion of an interrupt request when an edge-triggered event occurs on the $\overline{\text{IRQ}}[11]$ pin. 0 Interrupt request is disabled. 1 Interrupt request is enabled.
21 EIRE10	Interrupt request enable 10. This bit enables the assertion of an interrupt request when an edge-triggered event occurs on the $\overline{\text{IRQ}}[10]$ pin.

Table continues on the next page...

SIU_DIRER field descriptions (continued)

Field	Description
	0 Interrupt request is disabled. 1 Interrupt request is enabled.
22 EIRE9	Interrupt request enable 9. This bit enables the assertion of an interrupt request when an edge-triggered event occurs on the $\overline{\text{IRQ}}[9]$ pin. 0 Interrupt request is disabled. 1 Interrupt request is enabled.
23 EIRE8	Interrupt request enable 8. This bit enables the assertion of an interrupt request when an edge-triggered event occurs on the $\overline{\text{IRQ}}[8]$ pin. 0 Interrupt request is disabled. 1 Interrupt request is enabled.
24 EIRE7	Interrupt request enable 7. This bit enables the assertion of an interrupt request when an edge-triggered event occurs on the $\overline{\text{IRQ}}[7]$ pin. 0 Interrupt request is disabled. 1 Interrupt request is enabled.
25 EIRE6	Interrupt request enable 6. This bit enables the assertion of an interrupt request when an edge-triggered event occurs on the $\overline{\text{IRQ}}[6]$ pin. 0 Interrupt request is disabled. 1 Interrupt request is enabled.
26 EIRE5	Interrupt request enable 5. This bit enables the assertion of an interrupt request when an edge-triggered event occurs on the $\overline{\text{IRQ}}[5]$ pin. 0 Interrupt request is disabled. 1 Interrupt request is enabled.
27 EIRE4	Interrupt request enable 4. This bit enables the assertion of an interrupt request when an edge-triggered event occurs on the $\overline{\text{IRQ}}[4]$ pin. 0 Interrupt request is disabled. 1 Interrupt request is enabled.
28 EIRE3	DMA/interrupt request enable 3. This bit enables the assertion of a DMA request or an interrupt request when an edge-triggered event occurs on the $\overline{\text{IRQ}}[3]$ pin. NOTE: EIRE[0:3] can optionally enable DMA requests instead of IRQs. 0 DMA and interrupt requests are disabled. 1 DMA or interrupt request is enabled.
29 EIRE2	DMA/interrupt request enable 2. This bit enables the assertion of a DMA request or an interrupt request when an edge-triggered event occurs on the $\overline{\text{IRQ}}[2]$ pin. NOTE: EIRE[0:3] can optionally enable DMA requests instead of IRQs. 0 DMA and interrupt requests are disabled. 1 DMA or interrupt request is enabled.
30 EIRE1	DMA/interrupt request enable 1. This bit enables the assertion of a DMA request or an interrupt request when an edge-triggered event occurs on the $\overline{\text{IRQ}}[1]$ pin. NOTE: EIRE[0:3] can optionally enable DMA requests instead of IRQs.

Table continues on the next page...

SIU_DIRER field descriptions (continued)

Field	Description
	0 DMA and interrupt requests are disabled. 1 DMA or interrupt request is enabled.
31 EIRE0	DMA/interrupt request enable 0. This bit enables the assertion of a DMA request or an interrupt request when an edge-triggered event occurs on the $\overline{\text{IRQ}}[0]$ pin. NOTE: EIRE[0:3] can optionally enable DMA requests instead of IRQs. 0 DMA and interrupt requests are disabled. 1 DMA or interrupt request is enabled.

8.2.6 DMA/Interrupt Request Select Register (SIU_DIRSR)

The SIU_DIRSR selects between DMA and interrupt requests for events on the $\overline{\text{IRQ}}[0]$ - $\overline{\text{IRQ}}[3]$. If the corresponding bits are set in SIU_EISR and SIU_DIRER, then the DMA/Interrupt Request Select bit (DIRS n) determines whether a DMA or interrupt request is asserted.

Address: 0h base + 1Ch offset = 1Ch

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0												DIRS3	DIRS2	DIRS1	DIRS0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

SIU_DIRSR field descriptions

Field	Description
0–27 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
28 DIRS3	DMA/interrupt select 3. Selects between a DMA or external interrupt request when an edge-triggered event occurs on $\overline{\text{IRQ}}[3]$ pin. 0 Interrupt request is selected. 1 DMA request to eDMA_A channel 51 is selected.

Table continues on the next page...

SIU_DIRSR field descriptions (continued)

Field	Description
29 DIRS2	DMA/interrupt select 2. Selects between a DMA or external interrupt request when an edge-triggered event occurs on the corresponding $\overline{\text{IRQ}}[2]$ pin. 0 Interrupt request is selected. 1 DMA request to eDMA_A channel 50 is selected.
30 DIRS1	DMA/interrupt select 1. Selects between a DMA or external interrupt request when an edge-triggered event occurs on the corresponding $\overline{\text{IRQ}}[1]$ pin. 0 Interrupt request is selected. 1 DMA request to eDMA_A channel 49 is selected.
31 DIRS0	DMA/interrupt select 0. Selects between a DMA or external interrupt request when an edge-triggered event occurs on the corresponding $\overline{\text{IRQ}}[0]$ pin. 0 Interrupt request is selected. 1 DMA request to eDMA_A channel 48 is selected.

8.2.7 Overrun Status Register (SIU_OSR)

The SIU_OSR flag bits indicate that an overrun has occurred. An overrun is generated if an event on an $\overline{\text{IRQ}}$ pin occurs while the corresponding flag bit is set in SIU_EISR. The overrun flag bit is set regardless of the state of the corresponding DMA/Interrupt Request Enable bit in SIU_DIRER or the corresponding Overrun Request Enable bit in SIU_ORER. The numbering of the Overrun Flag bits correspond to the numbering of the flag bits in SIU_EISR. The Overrun Flag bit remains set until cleared by software. Each Overrun Flag bit is cleared by writing 1. Writing 0 has no effect.

Address: 0h base + 20h offset = 20h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	OVF15	OVF14	OVF13	OVF12	OVF11	OVF10	OVF9	OVF8	OVF7	OVF6	OVF5	OVF4	OVF3	OVF2	OVF1	OVF0
W	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

SIU_OSR field descriptions

Field	Description
0–15 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
16 OVF15	Overrun flag 15. This bit is set when an interrupt overrun occurs on $\overline{\text{IRQ}}[15]$. Bit 31 (OVF0) is the overrun flag for $\overline{\text{IRQ}}[0]$; bit 16 (OVF15) is overrun flag for $\overline{\text{IRQ}}[15]$. 0 No overrun occurred. 1 An overrun occurred.
17 OVF14	Overrun flag 14. This bit is set when an interrupt overrun occurs on $\overline{\text{IRQ}}[14]$. Bit 31 (OVF0) is the overrun flag for $\overline{\text{IRQ}}[0]$; bit 16 (OVF15) is overrun flag for $\overline{\text{IRQ}}[15]$.

Table continues on the next page...

SIU_OSR field descriptions (continued)

Field	Description
	0 No overrun occurred. 1 An overrun occurred.
18 OVF13	Overrun flag 13. This bit is set when an interrupt overrun occurs on $\overline{\text{IRQ}}[13]$. Bit 31 (OVF0) is the overrun flag for $\overline{\text{IRQ}}[0]$; bit 16 (OVF15) is overrun flag for $\overline{\text{IRQ}}[15]$. 0 No overrun occurred. 1 An overrun occurred.
19 OVF12	Overrun flag 12. This bit is set when an interrupt overrun occurs on $\overline{\text{IRQ}}[12]$. Bit 31 (OVF0) is the overrun flag for $\overline{\text{IRQ}}[0]$; bit 16 (OVF15) is overrun flag for $\overline{\text{IRQ}}[15]$. 0 No overrun occurred. 1 An overrun occurred.
20 OVF11	Overrun flag 11. This bit is set when an interrupt overrun occurs on $\overline{\text{IRQ}}[11]$. Bit 31 (OVF0) is the overrun flag for $\overline{\text{IRQ}}[0]$; bit 16 (OVF15) is overrun flag for $\overline{\text{IRQ}}[15]$. 0 No overrun occurred. 1 An overrun occurred.
21 OVF10	Overrun flag 10. This bit is set when an interrupt overrun occurs on $\overline{\text{IRQ}}[10]$. Bit 31 (OVF0) is the overrun flag for $\overline{\text{IRQ}}[0]$; bit 16 (OVF15) is overrun flag for $\overline{\text{IRQ}}[15]$. 0 No overrun occurred. 1 An overrun occurred.
22 OVF9	Overrun flag 9. This bit is set when an interrupt overrun occurs on $\overline{\text{IRQ}}[9]$. Bit 31 (OVF0) is the overrun flag for $\overline{\text{IRQ}}[0]$; bit 16 (OVF15) is overrun flag for $\overline{\text{IRQ}}[15]$. 0 No overrun occurred. 1 An overrun occurred.
23 OVF8	Overrun flag 8. This bit is set when an interrupt overrun occurs on $\overline{\text{IRQ}}[8]$. Bit 31 (OVF0) is the overrun flag for $\overline{\text{IRQ}}[0]$; bit 16 (OVF15) is overrun flag for $\overline{\text{IRQ}}[15]$. 0 No overrun occurred. 1 An overrun occurred.
24 OVF7	Overrun flag 7. This bit is set when an interrupt overrun occurs on $\overline{\text{IRQ}}[7]$. Bit 31 (OVF0) is the overrun flag for $\overline{\text{IRQ}}[0]$; bit 16 (OVF15) is overrun flag for $\overline{\text{IRQ}}[15]$. 0 No overrun occurred. 1 An overrun occurred.
25 OVF6	Overrun flag 6. This bit is set when an interrupt overrun occurs on $\overline{\text{IRQ}}[6]$. Bit 31 (OVF0) is the overrun flag for $\overline{\text{IRQ}}[0]$; bit 16 (OVF15) is overrun flag for $\overline{\text{IRQ}}[15]$. 0 No overrun occurred. 1 An overrun occurred.
26 OVF5	Overrun flag 5. This bit is set when an interrupt overrun occurs on $\overline{\text{IRQ}}[5]$. Bit 31 (OVF0) is the overrun flag for $\overline{\text{IRQ}}[0]$; bit 16 (OVF15) is overrun flag for $\overline{\text{IRQ}}[15]$. 0 No overrun occurred. 1 An overrun occurred.
27 OVF4	Overrun flag 4. This bit is set when an interrupt overrun occurs on $\overline{\text{IRQ}}[4]$. Bit 31 (OVF0) is the overrun flag for $\overline{\text{IRQ}}[0]$; bit 16 (OVF15) is overrun flag for $\overline{\text{IRQ}}[15]$.

Table continues on the next page...

SIU_OSR field descriptions (continued)

Field	Description
	0 No overrun occurred. 1 An overrun occurred.
28 OVF3	Overrun flag 3. This bit is set when a DMA or interrupt overrun occurs on $\overline{\text{IRQ}}[3]$. Bit 31 (OVF0) is the overrun flag for $\overline{\text{IRQ}}[0]$; bit 16 (OVF15) is overrun flag for $\overline{\text{IRQ}}[15]$. 0 No overrun occurred. 1 An overrun occurred.
29 OVF2	Overrun flag 2. This bit is set when a DMA or interrupt overrun occurs on $\overline{\text{IRQ}}[2]$. Bit 31 (OVF0) is the overrun flag for $\overline{\text{IRQ}}[0]$; bit 16 (OVF15) is overrun flag for $\overline{\text{IRQ}}[15]$. 0 No overrun occurred. 1 An overrun occurred.
30 OVF1	Overrun flag 1. This bit is set when a DMA or interrupt overrun occurs on $\overline{\text{IRQ}}[1]$. Bit 31 (OVF0) is the overrun flag for $\overline{\text{IRQ}}[0]$; bit 16 (OVF15) is overrun flag for $\overline{\text{IRQ}}[15]$. 0 No overrun occurred. 1 An overrun occurred.
31 OVF0	Overrun flag 0. This bit is set when a DMA or interrupt overrun occurs on $\overline{\text{IRQ}}[0]$. Bit 31 (OVF0) is the overrun flag for $\overline{\text{IRQ}}[0]$; bit 16 (OVF15) is overrun flag for $\overline{\text{IRQ}}[15]$. 0 No overrun occurred. 1 An overrun occurred.

8.2.8 Overrun Request Enable Register (SIU_ORER)

The SIU_ORER contains bits to enable an overrun if the corresponding flag bit is set in the SIU_OSR. If the overrun request enable bit and the flag bit are set, the single combined overrun request from the SIU to the interrupt controller is asserted.

Address: 0h base + 24h offset = 24h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	ORE15	ORE14	ORE13	ORE12	ORE11	ORE10	ORE9	ORE8	ORE7	ORE6	ORE5	ORE4	ORE3	ORE2	ORE1	ORE0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

SIU_OREr field descriptions

Field	Description
0–15 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
16 ORE15	Overflow request enable 15. Enables the overflow request when an overflow occurs in the $\overline{\text{IRQ}}[15]$ pin. Bit 31 (ORE0) is the enable overflow flag for $\overline{\text{IRQ}}[0]$; bit 16 (ORE15) is the overflow flag for $\overline{\text{IRQ}}[15]$. 0 Overflow request is disabled. 1 Overflow request is enabled.
17 ORE14	Overflow request enable 14. Enables the overflow request when an overflow occurs in the $\overline{\text{IRQ}}[14]$ pin. Bit 31 (ORE0) is the enable overflow flag for $\overline{\text{IRQ}}[0]$; bit 16 (ORE15) is the overflow flag for $\overline{\text{IRQ}}[15]$. 0 Overflow request is disabled. 1 Overflow request is enabled.
18 ORE13	Overflow request enable 13. Enables the overflow request when an overflow occurs in the $\overline{\text{IRQ}}[13]$ pin. Bit 31 (ORE0) is the enable overflow flag for $\overline{\text{IRQ}}[0]$; bit 16 (ORE15) is the overflow flag for $\overline{\text{IRQ}}[15]$. 0 Overflow request is disabled. 1 Overflow request is enabled.
19 ORE12	Overflow request enable 12. Enables the overflow request when an overflow occurs in the $\overline{\text{IRQ}}[12]$ pin. Bit 31 (ORE0) is the enable overflow flag for $\overline{\text{IRQ}}[0]$; bit 16 (ORE15) is the overflow flag for $\overline{\text{IRQ}}[15]$. 0 Overflow request is disabled. 1 Overflow request is enabled.
20 ORE11	Overflow request enable 11. Enables the overflow request when an overflow occurs in the $\overline{\text{IRQ}}[11]$ pin. Bit 31 (ORE0) is the enable overflow flag for $\overline{\text{IRQ}}[0]$; bit 16 (ORE15) is the overflow flag for $\overline{\text{IRQ}}[15]$. 0 Overflow request is disabled. 1 Overflow request is enabled.
21 ORE10	Overflow request enable 10. Enables the overflow request when an overflow occurs in the $\overline{\text{IRQ}}[10]$ pin. Bit 31 (ORE0) is the enable overflow flag for $\overline{\text{IRQ}}[0]$; bit 16 (ORE15) is the overflow flag for $\overline{\text{IRQ}}[15]$. 0 Overflow request is disabled. 1 Overflow request is enabled.
22 ORE9	Overflow request enable 9. Enables the overflow request when an overflow occurs in the $\overline{\text{IRQ}}[9]$ pin. Bit 31 (ORE0) is the enable overflow flag for $\overline{\text{IRQ}}[0]$; bit 16 (ORE15) is the overflow flag for $\overline{\text{IRQ}}[15]$. 0 Overflow request is disabled. 1 Overflow request is enabled.
23 ORE8	Overflow request enable 8. Enables the overflow request when an overflow occurs in the $\overline{\text{IRQ}}[8]$ pin. Bit 31 (ORE0) is the enable overflow flag for $\overline{\text{IRQ}}[0]$; bit 16 (ORE15) is the overflow flag for $\overline{\text{IRQ}}[15]$. 0 Overflow request is disabled. 1 Overflow request is enabled.
24 ORE7	Overflow request enable 7. Enables the overflow request when an overflow occurs in the $\overline{\text{IRQ}}[7]$ pin. Bit 31 (ORE0) is the enable overflow flag for $\overline{\text{IRQ}}[0]$; bit 16 (ORE15) is the overflow flag for $\overline{\text{IRQ}}[15]$. 0 Overflow request is disabled. 1 Overflow request is enabled.
25 ORE6	Overflow request enable 6. Enables the overflow request when an overflow occurs in the $\overline{\text{IRQ}}[6]$ pin. Bit 31 (ORE0) is the enable overflow flag for $\overline{\text{IRQ}}[0]$; bit 16 (ORE15) is the overflow flag for $\overline{\text{IRQ}}[15]$.

Table continues on the next page...

SIU_ORER field descriptions (continued)

Field	Description
	0 Overrun request is disabled. 1 Overrun request is enabled.
26 ORE5	Overrun request enable 5. Enables the overrun request when an overrun occurs in the $\overline{\text{IRQ}}[5]$ pin. Bit 31 (ORE0) is the enable overrun flag for $\overline{\text{IRQ}}[0]$; bit 16 (ORE15) is the overrun flag for $\overline{\text{IRQ}}[15]$. 0 Overrun request is disabled. 1 Overrun request is enabled.
27 ORE4	Overrun request enable 4. Enables the overrun request when an overrun occurs in the $\overline{\text{IRQ}}[4]$ pin. Bit 31 (ORE0) is the enable overrun flag for $\overline{\text{IRQ}}[0]$; bit 16 (ORE15) is the overrun flag for $\overline{\text{IRQ}}[15]$. 0 Overrun request is disabled. 1 Overrun request is enabled.
28 ORE3	Overrun request enable 3. Enables the overrun request when an overrun occurs in the $\overline{\text{IRQ}}[3]$ pin. Bit 31 (ORE0) is the enable overrun flag for $\overline{\text{IRQ}}[0]$; bit 16 (ORE15) is the overrun flag for $\overline{\text{IRQ}}[15]$. 0 Overrun request is disabled. 1 Overrun request is enabled.
29 ORE2	Overrun request enable 2. Enables the overrun request when an overrun occurs in the $\overline{\text{IRQ}}[2]$ pin. Bit 31 (ORE0) is the enable overrun flag for $\overline{\text{IRQ}}[0]$; bit 16 (ORE15) is the overrun flag for $\overline{\text{IRQ}}[15]$. 0 Overrun request is disabled. 1 Overrun request is enabled.
30 ORE1	Overrun request enable 1. Enables the overrun request when an overrun occurs in the $\overline{\text{IRQ}}[1]$ pin. Bit 31 (ORE0) is the enable overrun flag for $\overline{\text{IRQ}}[0]$; bit 16 (ORE15) is the overrun flag for $\overline{\text{IRQ}}[15]$. 0 Overrun request is disabled. 1 Overrun request is enabled.
31 ORE0	Overrun request enable 0. Enables the overrun request when an overrun occurs in the $\overline{\text{IRQ}}[0]$ pin. Bit 31 (ORE0) is the enable overrun flag for $\overline{\text{IRQ}}[0]$; bit 16 (ORE15) is the overrun flag for $\overline{\text{IRQ}}[15]$. 0 Overrun request is disabled. 1 Overrun request is enabled.

8.2.9 IRQ Rising-Edge Event Enable Register (SIU_IREER)

The SIU_IREER enables rising edge-triggered events on $\overline{\text{IRQ}}[n]$. Rising- and falling-edge events are enabled by setting the bits in SIU_IREER and SIU_IFEER.

SIU_IREER bits used for NMI events are write once after a reset.

Address: 0h base + 28h offset = 28h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R																
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Memory Map and Register Definition

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R																
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

SIU_IREE field descriptions

Field	Description
0 IREE_NMI8	Core 0 NMI rising-edge event enable. Enables rising edge triggered events on the external NMI pin to Core 0. This bit can be written once after a reset. 0 Rising edge event is disabled. 1 Rising edge event is enabled.
1 IREE_NMI7	Core 1 NMI rising-edge event enable. Enables rising edge triggered events on the external NMI pin to Core 1. This bit can be written once after a reset. 0 Rising edge event is disabled. 1 Rising edge event is enabled.
2–15 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
16 IREE15	IRQ rising-edge event enable 15. Enables rising edge triggered events on the external $\overline{\text{IRQ}}[15]$ pin. 0 Rising edge event is disabled. 1 Rising edge event is enabled.
17 IREE14	IRQ rising-edge event enable 14. Enables rising edge triggered events on the external $\overline{\text{IRQ}}[14]$ pin. 0 Rising edge event is disabled. 1 Rising edge event is enabled.
18 IREE13	IRQ rising-edge event enable 13. Enables rising edge triggered events on the external $\overline{\text{IRQ}}[13]$ pin. 0 Rising edge event is disabled. 1 Rising edge event is enabled.
19 IREE12	IRQ rising-edge event enable 12. Enables rising edge triggered events on the external $\overline{\text{IRQ}}[12]$ pin. 0 Rising edge event is disabled. 1 Rising edge event is enabled.
20 IREE11	IRQ rising-edge event enable 11. Enables rising edge triggered events on the external $\overline{\text{IRQ}}[11]$ pin. 0 Rising edge event is disabled. 1 Rising edge event is enabled.
21 IREE10	IRQ rising-edge event enable 10. Enables rising edge triggered events on the external $\overline{\text{IRQ}}[10]$ pin. 0 Rising edge event is disabled. 1 Rising edge event is enabled.
22 IREE9	IRQ rising-edge event enable 9. Enables rising edge triggered events on the external $\overline{\text{IRQ}}[9]$ pin. 0 Rising edge event is disabled. 1 Rising edge event is enabled.

Table continues on the next page...

SIU_IREER field descriptions (continued)

Field	Description
23 IREE8	IRQ rising-edge event enable 8. Enables rising edge triggered events on the external $\overline{\text{IRQ}}[8]$ pin. 0 Rising edge event is disabled. 1 Rising edge event is enabled.
24 IREE7	IRQ rising-edge event enable 7. Enables rising edge triggered events on the external $\overline{\text{IRQ}}[7]$ pin. 0 Rising edge event is disabled. 1 Rising edge event is enabled.
25 IREE6	IRQ rising-edge event enable 6. Enables rising edge triggered events on the external $\overline{\text{IRQ}}[6]$ pin. 0 Rising edge event is disabled. 1 Rising edge event is enabled.
26 IREE5	IRQ rising-edge event enable 5. Enables rising edge triggered events on the external $\overline{\text{IRQ}}[5]$ pin. 0 Rising edge event is disabled. 1 Rising edge event is enabled.
27 IREE4	IRQ rising-edge event enable 4. Enables rising edge triggered events on the external $\overline{\text{IRQ}}[4]$ pin. 0 Rising edge event is disabled. 1 Rising edge event is enabled.
28 IREE3	IRQ rising-edge event enable 3. Enables rising edge triggered events on the external $\overline{\text{IRQ}}[3]$ pin. 0 Rising edge event is disabled. 1 Rising edge event is enabled.
29 IREE2	IRQ rising-edge event enable 2. Enables rising edge triggered events on the external $\overline{\text{IRQ}}[2]$ pin. 0 Rising edge event is disabled. 1 Rising edge event is enabled.
30 IREE1	IRQ rising-edge event enable 1. Enables rising edge triggered events on the external $\overline{\text{IRQ}}[1]$ pin. 0 Rising edge event is disabled. 1 Rising edge event is enabled.
31 IREE0	IRQ rising-edge event enable 0. Enables rising edge triggered events on the external $\overline{\text{IRQ}}[0]$ pin. 0 Rising edge event is disabled. 1 Rising edge event is enabled.

8.2.10 IRQ Falling-Edge Event Enable Register (SIU_IFEER)

The SIU_IFEER enables falling edge-triggered events on $\overline{\text{IRQ}}[n]$. Rising- and falling-edge events are enabled by setting the bits in both SIU_IREER and SIU_IFEER.

SIU_IFEER bits used for NMI events are write once after a reset.

Memory Map and Register Definition

Address: 0h base + 2Ch offset = 2Ch

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	IFEE_NMI8	IFEE_NMI7	0													
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	IFEEN15	IFEEN14	IFEEN13	IFEEN12	IFEEN11	IFEEN10	IFEEN9	IFEEN8	IFEEN7	IFEEN6	IFEEN5	IFEEN4	IFEEN3	IFEEN2	IFEEN1	IFEEN0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

SIU_IFEER field descriptions

Field	Description
0 IFEE_NMI8	Core 0 NMI falling-edge event enable. Enables falling edge triggered events on the external NMI pin to Core 0. This bit can be written once after a reset. 0 Falling edge event is disabled. 1 Falling edge event is enabled.
1 IFEE_NMI7	Core 1 NMI falling-edge event enable. Enables falling edge triggered events on the external NMI pin to Core 1. This bit can be written once after a reset. 0 Falling edge event is disabled. 1 Falling edge event is enabled.
2–15 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
16 IFEEN15	IRQ falling-edge event enable 15. Enables falling edge triggered events on the external $\overline{\text{IRQ}}[15]$ pin. 0 Falling edge event is disabled. 1 Falling edge event is enabled.
17 IFEEN14	IRQ falling-edge event enable 14. Enables falling edge triggered events on the external $\overline{\text{IRQ}}[14]$ pin. 0 Falling edge event is disabled. 1 Falling edge event is enabled.
18 IFEEN13	IRQ falling-edge event enable 13. Enables falling edge triggered events on the external $\overline{\text{IRQ}}[13]$ pin. 0 Falling edge event is disabled. 1 Falling edge event is enabled.
19 IFEEN12	IRQ falling-edge event enable 12. Enables falling edge triggered events on the external $\overline{\text{IRQ}}[12]$ pin. 0 Falling edge event is disabled. 1 Falling edge event is enabled.
20 IFEEN11	IRQ falling-edge event enable 11. Enables falling edge triggered events on the external $\overline{\text{IRQ}}[11]$ pin. 0 Falling edge event is disabled. 1 Falling edge event is enabled.

Table continues on the next page...

SIU_IFEER field descriptions (continued)

Field	Description
21 IFEEN10	IRQ falling-edge event enable 10. Enables falling edge triggered events on the external $\overline{\text{IRQ}}[10]$ pin. 0 Falling edge event is disabled. 1 Falling edge event is enabled.
22 IFEEN9	IRQ falling-edge event enable 9. Enables falling edge triggered events on the external $\overline{\text{IRQ}}[9]$ pin. 0 Falling edge event is disabled. 1 Falling edge event is enabled.
23 IFEEN8	IRQ falling-edge event enable 8. Enables falling edge triggered events on the external $\overline{\text{IRQ}}[8]$ pin. 0 Falling edge event is disabled. 1 Falling edge event is enabled.
24 IFEEN7	IRQ falling-edge event enable 7. Enables falling edge triggered events on the external $\overline{\text{IRQ}}[7]$ pin. 0 Falling edge event is disabled. 1 Falling edge event is enabled.
25 IFEEN6	IRQ falling-edge event enable 6. Enables falling edge triggered events on the external $\overline{\text{IRQ}}[6]$ pin. 0 Falling edge event is disabled. 1 Falling edge event is enabled.
26 IFEEN5	IRQ falling-edge event enable 5. Enables falling edge triggered events on the external $\overline{\text{IRQ}}[5]$ pin. 0 Falling edge event is disabled. 1 Falling edge event is enabled.
27 IFEEN4	IRQ falling-edge event enable 4. Enables falling edge triggered events on the external $\overline{\text{IRQ}}[4]$ pin. 0 Falling edge event is disabled. 1 Falling edge event is enabled.
28 IFEEN3	IRQ falling-edge event enable 3. Enables falling edge triggered events on the external $\overline{\text{IRQ}}[3]$ pin. 0 Falling edge event is disabled. 1 Falling edge event is enabled.
29 IFEEN2	IRQ falling-edge event enable 2. Enables falling edge triggered events on the external $\overline{\text{IRQ}}[2]$ pin. 0 Falling edge event is disabled. 1 Falling edge event is enabled.
30 IFEEN1	IRQ falling-edge event enable 1. Enables falling edge triggered events on the external $\overline{\text{IRQ}}[1]$ pin. 0 Falling edge event is disabled. 1 Falling edge event is enabled.
31 IFEEN0	IRQ falling-edge event enable 0. Enables falling edge triggered events on the external $\overline{\text{IRQ}}[0]$ pin. 0 Falling edge event is disabled. 1 Falling edge event is enabled.

8.2.11 IRQ Digital Filter Register (SIU_IDFR)

The SIU_IDFR specifies the amount of digital filtering on $\overline{\text{IRQ}}[0]-\overline{\text{IRQ}}[15]$ and NMI. The digital filter length field specifies the number of system clocks that define the period of the digital filter and the minimum time an IRQ or NMI signal must hold the active state to qualify as an edge-triggered event.

Address: 0h base + 30h offset = 30h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0																											DFL				
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

SIU_IDFR field descriptions

Field	Description
0–27 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
28–31 DFL	Digital filter length. Defines the digital filter period on the $\overline{\text{IRQ}}[n]$ and NMI inputs according to the following equation: $\text{FilterPeriod} = (\text{PeripheralClockPeriod} \times 2^{\text{DFL}}) + 1(\text{PeripheralClockPeriod})$ For a 100 MHz peripheral clock, this gives a range of 20 ns to 328 ns. The minimum time of two system clocks accounts for synchronization of the $\overline{\text{IRQ}}$ and NMI input pins with the system clock.

8.2.12 IRQ Filtered Input Register (SIU_IFIR)

The SIU_IFIR is a read only register where the filtered values of the NMI and $\overline{\text{IRQ}}[0]-\overline{\text{IRQ}}[15]$ pins are captured.

Address: 0h base + 34h offset = 34h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	IFI_NMI8	IFI_NMI7	0													
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	IFI15	IFI14	IFI13	IFI12	IFI11	IFI10	IFI9	IFI8	IFI7	IFI6	IFI5	IFI4	IFI3	IFI2	IFI1	IFI0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

SIU_IFIR field descriptions

Field	Description
0 IFI_NMI8	NMI Filtered Input. This bit is set/cleared by the filtered NMI interrupt input. 0 A logic one has passed through the digital filter of the corresponding NMI pin. 1 A logic zero has passed through the digital filter of the corresponding NMI pin.
1 IFI_NMI7	NMI Filtered Input. This bit is set/cleared by the filtered NMI interrupt input. 0 A logic one has passed through the digital filter of the corresponding NMI pin. 1 A logic zero has passed through the digital filter of the corresponding NMI pin.
2–15 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
16 IFI15	Filtered Input 15. This bit is set/cleared for the corresponding filtered $\overline{\text{IRQ}}$ pin. 0 A logic one has passed through the digital filter of the corresponding $\overline{\text{IRQ}}$ pin. 1 A logic zero has passed through the digital filter of the corresponding $\overline{\text{IRQ}}$ pin.
17 IFI14	Filtered Input 14. This bit is set/cleared for the corresponding filtered $\overline{\text{IRQ}}$ pin. 0 A logic one has passed through the digital filter of the corresponding $\overline{\text{IRQ}}$ pin. 1 A logic zero has passed through the digital filter of the corresponding $\overline{\text{IRQ}}$ pin.
18 IFI13	Filtered Input 13. This bit is set/cleared for the corresponding filtered $\overline{\text{IRQ}}$ pin. 0 A logic one has passed through the digital filter of the corresponding $\overline{\text{IRQ}}$ pin. 1 A logic zero has passed through the digital filter of the corresponding $\overline{\text{IRQ}}$ pin.
19 IFI12	Filtered Input 12. This bit is set/cleared for the corresponding filtered $\overline{\text{IRQ}}$ pin. 0 A logic one has passed through the digital filter of the corresponding $\overline{\text{IRQ}}$ pin. 1 A logic zero has passed through the digital filter of the corresponding $\overline{\text{IRQ}}$ pin.
20 IFI11	Filtered Input 11. This bit is set/cleared for the corresponding filtered $\overline{\text{IRQ}}$ pin. 0 A logic one has passed through the digital filter of the corresponding $\overline{\text{IRQ}}$ pin. 1 A logic zero has passed through the digital filter of the corresponding $\overline{\text{IRQ}}$ pin.
21 IFI10	Filtered Input 10. This bit is set/cleared for the corresponding filtered $\overline{\text{IRQ}}$ pin. 0 A logic one has passed through the digital filter of the corresponding $\overline{\text{IRQ}}$ pin. 1 A logic zero has passed through the digital filter of the corresponding $\overline{\text{IRQ}}$ pin.
22 IFI9	Filtered Input 9. This bit is set/cleared for the corresponding filtered $\overline{\text{IRQ}}$ pin. 0 A logic one has passed through the digital filter of the corresponding $\overline{\text{IRQ}}$ pin. 1 A logic zero has passed through the digital filter of the corresponding $\overline{\text{IRQ}}$ pin.
23 IFI8	Filtered Input 8. This bit is set/cleared for the corresponding filtered $\overline{\text{IRQ}}$ pin. 0 A logic one has passed through the digital filter of the corresponding $\overline{\text{IRQ}}$ pin. 1 A logic zero has passed through the digital filter of the corresponding $\overline{\text{IRQ}}$ pin.
24 IFI7	Filtered Input 7. This bit is set/cleared for the corresponding filtered $\overline{\text{IRQ}}$ pin. 0 A logic one has passed through the digital filter of the corresponding $\overline{\text{IRQ}}$ pin. 1 A logic zero has passed through the digital filter of the corresponding $\overline{\text{IRQ}}$ pin.
25 IFI6	Filtered Input 6. This bit is set/cleared for the corresponding filtered $\overline{\text{IRQ}}$ pin.

Table continues on the next page...

SIU_IFIR field descriptions (continued)

Field	Description
	0 A logic one has passed through the digital filter of the corresponding $\overline{\text{IRQ}}$ pin. 1 A logic zero has passed through the digital filter of the corresponding $\overline{\text{IRQ}}$ pin.
26 IFI5	Filtered Input 5. This bit is set/cleared for the corresponding filtered $\overline{\text{IRQ}}$ pin. 0 A logic one has passed through the digital filter of the corresponding $\overline{\text{IRQ}}$ pin. 1 A logic zero has passed through the digital filter of the corresponding $\overline{\text{IRQ}}$ pin.
27 IFI4	Filtered Input 4. This bit is set/cleared for the corresponding filtered $\overline{\text{IRQ}}$ pin. 0 A logic one has passed through the digital filter of the corresponding $\overline{\text{IRQ}}$ pin. 1 A logic zero has passed through the digital filter of the corresponding $\overline{\text{IRQ}}$ pin.
28 IFI3	Filtered Input 3. This bit is set/cleared for the corresponding filtered $\overline{\text{IRQ}}$ pin. 0 A logic one has passed through the digital filter of the corresponding $\overline{\text{IRQ}}$ pin. 1 A logic zero has passed through the digital filter of the corresponding $\overline{\text{IRQ}}$ pin.
29 IFI2	Filtered Input 2. This bit is set/cleared for the corresponding filtered $\overline{\text{IRQ}}$ pin. 0 A logic one has passed through the digital filter of the corresponding $\overline{\text{IRQ}}$ pin. 1 A logic zero has passed through the digital filter of the corresponding $\overline{\text{IRQ}}$ pin.
30 IFI1	Filtered Input 1. This bit is set/cleared for the corresponding filtered $\overline{\text{IRQ}}$ pin. 0 A logic one has passed through the digital filter of the corresponding $\overline{\text{IRQ}}$ pin. 1 A logic zero has passed through the digital filter of the corresponding $\overline{\text{IRQ}}$ pin.
31 IFI0	Filtered Input 0. This bit is set/cleared for the corresponding filtered $\overline{\text{IRQ}}$ pin. 0 A logic one has passed through the digital filter of the corresponding $\overline{\text{IRQ}}$ pin. 1 A logic zero has passed through the digital filter of the corresponding $\overline{\text{IRQ}}$ pin.

8.2.13 Pad Configuration Register (SIU_PCRn)

These registers define the pad configuration for all configurable device pins by specifying the active function, direction, and electrical attributes for the pin. PCR information includes the register fields that are active for a given pin or group of pins and the reset state of the register.

The reset states of SIU_PCRs reflect the values before the BAM program executes. The BAM program can change some pad configuration registers based on the reset configuration. Refer to the System Boot chapter for more information.

The SIU_PCRs are 16-bit registers that are read from or written to as:

- 16-bit values aligned on 16-bit boundaries, or
- 32-bit values aligned on 32-bit address boundaries.

The PCR registers allow software to control the following range of electrical characteristics of the external pads:

- Weak pullup/down enable/disable
- Weak pullup/down selection
- Slew-rate selection for outputs
- Drive strength selection for outputs
- Input buffer enable (when direction is configured for output)
- Input hysteresis enable/disable
- Open drain/push-pull output selection
- Multiplexed function selection
- Data direction selection

NOTE

The actual characteristics available in any SIU_PCR depend on the type of pad it controls.

All device pin names begin with the primary function, followed by the alternate function, and then GPIO. In some cases, the third function can be a secondary alternate, which supersedes the GPIO. Those exceptions are noted. For example, SIU_PCR85 configures the CNTXB_PCSC[3]_GPIO[85] muxed signal, where CNTXB is the primary function and PCSC[3] is the alternate function. See [External signal descriptions, multiplexing, and attributes](#) for how to identify the source module for primary and alternate functions and the description of these signals. For an additional signal description, see the chapter for the module that uses this signal.

A number of input signals on the device have multiple external pins as input sources. Only one input source can be active at any time for these pins. A set of multiplexers controlled by the SIU_IMUX0 to SIU_IMUX12 registers selects which input drives the function.

For information about implemented fields and reset states of individual PCR_s, see [Pad Configuration Register \(PCR\) details](#).

Address: 0h base + D6h offset + (2d × i), where i=0d to 416d

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Read	0			PA			OBE	IBE	DSC		ODE	HYS	SRC		WPE	WPS
Write																
Reset	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*

* Notes:

- See [Table 8-3](#) for reset values

SIU_PCR_n field descriptions

Field	Description
0–2 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

Table continues on the next page...

SIU_PCR_n field descriptions (continued)

Field	Description
3–5 PA	Pin assignment. Selects the function of a multiplexed pad. A separate port enable output signal from the SIU is asserted for each value of this register. The size of the field can be from 1 to 3 bits, depending on the amount of multiplexing on the pad. NOTE: For any SIU_PCR that does not comply with these rules, the PA definition is given explicitly with the SIU_PCR definition. 000 GPIO. 001 Primary function. 010 Alternate 1. 011 Alternate 2. 100 Alternate 3. 101 Invalid value. 110 Invalid value. 111 Invalid value.
6 OBE	Output buffer enable. Enables the pad as an output and drives the output buffer enable signal. 0 Disable output buffer for the pad. 1 Enable output buffer for the pad.
7 IBE	Input buffer enable. Enables the pad as an input and drives the input buffer enable signal. 0 Disable input buffer for the pad. 1 Enable input buffer for the pad.
8–9 DSC	Drive strength control. Controls the pad drive strength. Actual drive strength depends on the pad type and load. See the electrical specifications in the Data Sheet for this information. 00 Minimum drive strength. 01 Medium drive strength. 10 Medium-high drive strength. 11 Maximum drive strength.
10 ODE	Open drain output enable. Controls output driver configuration for the pads. Either open drain or push/pull driver configurations can be selected. This feature applies to output pins only. 0 Disable open drain for the pad (push/pull driver enabled). 1 Enable open drain for the pad.
11 HYS	Input hysteresis. Controls whether hysteresis is enabled for the pad. 0 Disable hysteresis for the pad. 1 Enable hysteresis for the pad.
12–13 SRC	Slew rate control. Controls slew rate for the pad. Slew rate control pertains to pins with slow or medium I/O pad types, and the output signals are driven according to the value of this field. Actual slew rate depends on the pad type and load. See the electrical specifications in the Data Sheet for this information. 00 Half drive strength with slew control enabled. 01 Full drive strength with slew control enabled. 10 Half drive strength with slew control disabled. 11 Full drive strength with slew control disabled.
14 WPE	Weak pull up/down enable. Controls whether the weak pullup/down devices are enabled/disabled for the pad. Pullup/down devices are enabled by default.

Table continues on the next page...

SIU_PCR_n field descriptions (continued)

Field	Description
	0 Disable weak pull device for the pad. 1 Enable weak pull device for the pad.
15 WPS	Weak pull up/down select. Controls whether weak pullup or weak pulldown devices are used for the pad when weak pullup/down devices are enabled. During reset, the weak pull up/down behavior is controlled by the WKPCFG bit of the Miscellaneous DCF client, or by the WKPCFG pin when operating in legacy mode ((LEG=1 in the Miscellaneous DCF client). The WPS bit determines whether weak pullup or pulldown devices are used after reset, or for pads in which the WKPCFG pin does not determine the reset weak pullup/down state. 0 Pulldown is enabled for the pad. 1 Pullup is enabled for the pad.

8.2.14 GPIO Pin Data Output Register (SIU_GPDO_n)

Each SIU_GPDO_n register specifies the output data for the function assigned to the GPIO[*n*] pin. The *n* notation in the SIU_GPDO_n register name relate to the [*n*] in GPIO[*n*] signal name. For example, SIU_GPDO246 contains the PDO246 bit for CNTXD_GPIO246. The address for a GPDO pin is the GPIO number plus an offset of SIU_BASE + 0600h. Gaps exist in the memory map where registers correspond to GPIO pins that are not implemented in the package.

Software writes to the SIU_GPDO_n registers to drive data out on the external pin. Each register drives one external pin, which allows independent control of the pin. Writes to the SIU_GPDO_n register will not be reflected to the pin when a primary or alternate function is assigned, or when an input function is assigned to the pin.

If the direction of a GPIO signal changes from input to output, the SIU_GPDO_n register value is automatically driven out to the external pin without the need for a software update.

Writes to an SIU_GPDO_n register have no effect when a primary or alternate function is assigned.

Address: 0h base + 64Bh offset + (1d × *i*), where *i*=0d to 416d

Bit	0	1	2	3	4	5	6	7
Read	0						PDO _n	
Write								
Reset	0	0	0	0	0	0	0	0

SIU_GPDO_n field descriptions

Field	Description
0–6 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

Table continues on the next page...

SIU_GPDO_n field descriptions (continued)

Field	Description
7 PDO _n	Pin Data Out. This bit stores the data to be driven out on the external GPIO. If the register is read, it returns the value written. 0 Logic low value is driven to the external GPIO pin when the pin is configured as an output. 1 Logic high value is driven to the external GPIO pin when the pin is configured as an output.

8.2.15 GPIO Pin Data Input Register (legacy) (SIU_GPDInL)**NOTE**

These registers are implemented for legacy purposes and are limited to 256 general purpose input registers. The GPDI address for a particular pin is the GPIO number plus an offset of SIU_BASE + 0800h. For full access to all 512 general purpose input registers, the SIU_GPDIn registers starting at address offset 0E00h should be used.

Each SIU_GPDIn register specifies the input state for the GPDI[*n*] pin. The *n* notation in the SIU_GPDIn register name relates to the [*n*] in GPIO[*n*] signal name. For example, SIU_GPDI246 contains the PDI246 bit for CNTXD_GPIO246. Gaps exist in the memory map where registers correspond to GPIO pins that are not implemented in the package.

Software reads the SIU_GPDIn registers to get the input state of the external GPIO pin. Each GPDI register contains the input state of one external GPIO pin. If a GPDI register is configured as output, and the input buffer enable bit is set to one in the PCR register, the SIU_GPDIn register reflects the state of the output pin.

Address: 0h base + 84Bh offset + (1d × i), where i=0d to 178d

Bit	0	1	2	3	4	5	6	7
Read	0						PDI _n	
Write								
Reset	0	0	0	0	0	0	0	0

SIU_GPDInL field descriptions

Field	Description
0–6 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
7 PDI _n	Pin Data In. This bit reflects the state on the external GPIO/GPI pin. If PCR _n [IBE] = 1, then: 0 Signal on pin is logic zero. 1 Signal on pin is logic one.

8.2.16 External IRQ Input Select Register (SIU_EIISR)

The SIU_EIISR selects the source for the external interrupt/DMA inputs.

Address: 0h base + 904h offset = 904h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R																
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R																
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

SIU_EIISR field descriptions

Field	Description
0–1 ESEL15	External IRQ input select 15. Specifies the source for $\overline{\text{IRQ}}[15]$. 00 $\overline{\text{IRQ}}[15]$ input pin. 01 DSPI_B[15] deserialized input. 10 DSPI_C[0] deserialized input. 11 DSPI_D[1] deserialized input.
2–3 ESEL14	External IRQ input select 14. Specifies the source for $\overline{\text{IRQ}}[14]$. 00 $\overline{\text{IRQ}}[14]$ input pin. 01 DSPI_B[14] deserialized input. 10 DSPI_C[15] deserialized input. 11 DSPI_D[0] deserialized input.
4–5 ESEL13	External IRQ input select 13. Specifies the source for $\overline{\text{IRQ}}[13]$. 00 $\overline{\text{IRQ}}[13]$ input pin. 01 DSPI_B[13] deserialized input. 10 DSPI_C[14] deserialized input. 11 DSPI_D[15] deserialized input.
6–7 ESEL12	External IRQ input select 12. Specifies the source for $\overline{\text{IRQ}}[12]$. 00 $\overline{\text{IRQ}}[12]$ input pin. 01 DSPI_B[12] deserialized input. 10 DSPI_C[13] deserialized input. 11 DSPI_D[14] deserialized input.
8–9 ESEL11	External IRQ input select 11. Specifies the source for $\overline{\text{IRQ}}[11]$. 00 $\overline{\text{IRQ}}[11]$ input pin. 01 DSPI_B[11] deserialized input. 10 DSPI_C[12] deserialized input. 11 DSPI_D[13] deserialized input.

Table continues on the next page...

SIU_EIISR field descriptions (continued)

Field	Description
10–11 ESEL10	External IRQ input select 10. Specifies the source for $\overline{\text{IRQ}}[10]$. 00 $\overline{\text{IRQ}}[10]$ input pin. 01 DSPI_B[10] deserialized input. 10 DSPI_C[11] deserialized input. 11 DSPI_D[12] deserialized input.
12–13 ESEL9	External IRQ input select 9. Specifies the source for $\overline{\text{IRQ}}[9]$. 00 $\overline{\text{IRQ}}[9]$ input pin. 01 DSPI_B[9] deserialized input. 10 DSPI_C[10] deserialized input. 11 DSPI_D[11] deserialized input.
14–15 ESEL8	External IRQ input select 8. Specifies the source for $\overline{\text{IRQ}}[8]$. 00 $\overline{\text{IRQ}}[8]$ input pin. 01 DSPI_B[8] deserialized input. 10 DSPI_C[9] deserialized input. 11 DSPI_D[10] deserialized input.
16–17 ESEL7	External IRQ input select 7. Specifies the source for $\overline{\text{IRQ}}[7]$. 00 $\overline{\text{IRQ}}[7]$ input pin. 01 DSPI_B[7] deserialized input. 10 DSPI_C[8] deserialized input. 11 DSPI_D[9] deserialized input.
18–19 ESEL6	External IRQ input select 6. Specifies the source for $\overline{\text{IRQ}}[6]$. 00 $\overline{\text{IRQ}}[6]$ input pin. 01 DSPI_B[6] deserialized input. 10 DSPI_C[7] deserialized input. 11 DSPI_D[8] deserialized input.
20–21 ESEL5	External IRQ input select 5. Specifies the source for $\overline{\text{IRQ}}[5]$. 00 $\overline{\text{IRQ}}[5]$ input pin. 01 DSPI_B[5] deserialized input. 10 DSPI_C[6] deserialized input. 11 DSPI_D[7] deserialized input.
22–23 ESEL4	External IRQ input select 4. Specifies the source for $\overline{\text{IRQ}}[4]$. 00 $\overline{\text{IRQ}}[4]$ input pin. 01 DSPI_B[4] deserialized input. 10 DSPI_C[5] deserialized input. 11 DSPI_D[6] deserialized input.
24–25 ESEL3	External IRQ input select 3. Specifies the source for $\overline{\text{IRQ}}[3]$. 00 $\overline{\text{IRQ}}[3]$ input pin. 01 DSPI_B[3] deserialized input. 10 DSPI_C[4] deserialized input. 11 DSPI_D[5] deserialized input.

Table continues on the next page...

SIU_EIISR field descriptions (continued)

Field	Description
26–27 ESEL2	External IRQ input select 2. Specifies the source for $\overline{\text{IRQ}}[2]$. 00 $\overline{\text{IRQ}}[2]$ input pin. 01 DSPI_B[2] deserialized input. 10 DSPI_C[3] deserialized input. 11 DSPI_D[4] deserialized input.
28–29 ESEL1	External IRQ input select 1. Specifies the source for $\overline{\text{IRQ}}[1]$. 00 $\overline{\text{IRQ}}[1]$ input pin. 01 DSPI_B[1] deserialized input. 10 DSPI_C[2] deserialized input. 11 eMIOS_1 channel 7 output.
30–31 ESEL0	External IRQ input select 0. Specifies the source for $\overline{\text{IRQ}}[0]$. 00 $\overline{\text{IRQ}}[0]$ input pin. 01 DSPI_B[0] deserialized input. 10 DSPI_C[1] deserialized input. 11 eMIOS_1 channel 6 output.

8.2.17 DSPI Input Select Register (SIU_DISR)

The SIU_DISR specifies the following operations for each DSPI:

- Data input source
- Slave select
- Clock input

Address: 0h base + 908h offset = 908h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R																
W																
	SINSELA		SSSELA		SCKSELA		Reserved		SINSELB		SSSELB		SCKSELB		Reserved	
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R																
W																
	SINSELC		SSSELC		SCKSELC		Reserved		SINSELD		SSSELD		SCKSELD		Reserved	
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

SIU_DISR field descriptions

Field	Description
0–1 SINSELA	DSPI A data input select. Specifies the source of the DSPI A data input. 00 SINA_PCSC[2]_GPIO[94] pin. 01 SOUTB.

Table continues on the next page...

SIU_DISR field descriptions (continued)

Field	Description
	10 SOUTC. 11 SOUTD.
2–3 SSSELA	DSPI A slave select input select. Specifies the source of the DSPI A slave select input. 00 PCSA[0]_PCSD[2]_GPIO[96] pin. 01 PCSB[0] (master). 10 PCSC[0] (master). 11 PCSD[0] (master).
4–5 SCKSELA	DSPI A clock input select. Specifies the source of the DSPI A clock input. 00 SCKA_PCSC[1]_GPIO[93] pin. 01 SCKB (master). 10 SCKC (master). 11 SCKD (master).
6–7 Reserved	This field is reserved. Always preserve this field's reset value.
8–9 SINSELB	DSPI B data input select. Specifies the source of DSPI B data input. 00 SINB_GPIO[103] pin. 01 SOUTA. 10 SOUTC. 11 SOUTD.
10–11 SSSELB	DSPI B slave select input select. Specifies the source of the DSPI B slave select input. 00 PCSB[0]_PCSD[2]_GPIO[105] pin. 01 PCSA[0] (master). 10 PCSC[0] (master). 11 PCSD[0] (master).
12–13 SCKSELB	DSPI B clock input select. Specifies the source of the DSPI B clock input. 00 SCKB_GPIO[102] pin. 01 SCKA (master). 10 SCKC (master). 11 SCKD (master).
14–15 Reserved	This field is reserved. Always preserve this field's reset value.
16–17 SINSELC	DSPI C data input select. Specifies the source of the DSPI C data input. 00 PCSB[3]_SINC_GPIO[108] pin. 01 SOUTA. 10 SOUTB. 11 SOUTD.
18–19 SSSELC	DSPI C slave select input select. Specifies the source of the DSPI C slave select input. 00 PCSB[5]_PCSC[0]_GPIO[110] pin. 01 PCSA[0] (master). 10 PCSB[0] (master). 11 PCSD[0] (master).

Table continues on the next page...

SIU_DISR field descriptions (continued)

Field	Description
20–21 SCKSELC	DSPI C clock input select. Specifies the source of the DSPI C clock input when in slave mode. 00 PCSB[4]_SCKC_GPIO[109] pin. 01 SCKA (master). 10 SCKB (master). 11 SCKD (master).
22–23 Reserved	This field is reserved. Always preserve this field's reset value.
24–25 SINSELD	DSPI D data input select. Specifies the source of the DSPI D data input. 00 PCSA[3]_GPIO[99] pin. 01 SOUTA. 10 SOUTB. 11 SOUTC.
26–27 SSSELD	DSPI D slave select input select. Specifies the source of the DSPI D slave select input. 00 PCSB[1]_PCSD[0]_GPIO[106] pin. 01 PCSA0 (master). 10 PCSB0 (master). 11 PCSC0 (master).
28–29 SCKSELD	DSPI D clock input select. Specifies the source of the DSPI D clock input in slave mode. 00 PCSA[2]_SCKD_GPIO[98] pin. 01 Invalid value. 10 SCKB (master). 11 SCKC (master).
30–31 Reserved	This field is reserved. Always preserve this field's reset value.

8.2.18 eQADC Command FIFO Trigger Source Registers (SIU_ISEL4)

The eQADC Command FIFO[5:2] Trigger Source Register (SIU_ISEL4) is used to select a trigger source for a command FIFO. SIU_ISEL4 is used to select a trigger source for command FIFO[5:2] in eQADC_A. The cTSEL (combined Trigger Select) field is used to configure one of many possible trigger sources for each command FIFO.

To trigger the eQADC, the trigger source must change to the state that the input to the command FIFO has been programmed to recognize. A command FIFO trigger input can be programmed to recognize either rising or falling edges, and low or high gated trigger types.

Table 8-1036. eQADC_A Command FIFO 2 Trigger Sources

cTSEL_2_A							eQADC_A Trigger Inputs
0	0	0	0	0	0	0	Not Connected (default)
0	0	0	0	0	0	1	RTI Trigger
0	0	0	0	0	1	0	PIT0 Trigger
0	0	0	0	0	1	1	PIT1 Trigger
0	0	0	0	1	0	0	PIT2 Trigger
0	0	0	0	1	0	1	PIT3 Trigger
0	0	0	0	1	1	0	eTPUA14
0	0	0	0	1	1	1	ETRIG1 pin
0	0	0	1	0	x	x	Reserved
0	0	0	1	1	0	0	eTPUA28
0	0	0	1	1	0	1	eTPUA29
0	0	0	1	1	1	0	eTPUA30
0	0	0	1	1	1	1	eTPUA31
0	0	1	0	0	0	0	eTPUB28
0	0	1	0	0	0	1	eTPUB29
0	0	1	0	0	1	0	eTPUB30
0	0	1	0	0	1	1	eTPUB31
0	0	1	0	1	0	0	eTPUC28
0	0	1	0	1	0	1	eTPUC29
0	0	1	0	1	1	0	eTPUC30
0	0	1	0	1	1	1	eTPUC31
0	0	1	1	0	0	0	eMIOS_0 channel 16
0	0	1	1	0	0	1	eMIOS_0 channel 17
0	0	1	1	0	1	0	eMIOS_0 channel 18
0	0	1	1	0	1	1	eMIOS_0 channel 19
0	0	1	1	1	0	0	eMIOS_0 channel 20
0	0	1	1	1	0	1	eMIOS_0 channel 21
0	0	1	1	1	1	0	eMIOS_0 channel 22
0	0	1	1	1	1	1	eMIOS_0 channel 23
0	1	x	x	x	x	x	eTPUA29
1	0	x	x	x	x	x	eMIOS_1 channel 7
1	1	x	x	x	x	x	ETRIG0 Pin

Table 8-1037. eQADC_A Command FIFO 3 Trigger Sources

cTSEL_3_A							eQADC_A Trigger Inputs
0	0	0	0	0	0	0	Not Connected (default)
0	0	0	0	0	0	1	RTI Trigger
0	0	0	0	0	1	0	PIT0 Trigger

Table continues on the next page...

Table 8-1037. eQADC_A Command FIFO 3 Trigger Sources (continued)

cTSEL_3_A							eQADC_A Trigger Inputs
0	0	0	0	0	1	1	PIT1 Trigger
0	0	0	0	1	0	0	PIT2 Trigger
0	0	0	0	1	0	1	PIT3 Trigger
0	0	0	0	1	1	0	eTPUA22
0	0	0	0	1	1	1	ETRIG0 pin
0	0	0	1	0	x	x	Reserved
0	0	0	1	1	0	0	eTPUA28
0	0	0	1	1	0	1	eTPUA29
0	0	0	1	1	1	0	eTPUA30
0	0	0	1	1	1	1	eTPUA31
0	0	1	0	0	0	0	eTPUB28
0	0	1	0	0	0	1	eTPUB29
0	0	1	0	0	1	0	eTPUB30
0	0	1	0	0	1	1	eTPUB31
0	0	1	0	1	0	0	eTPUC28
0	0	1	0	1	0	1	eTPUC29
0	0	1	0	1	1	0	eTPUC30
0	0	1	0	1	1	1	eTPUC31
0	0	1	1	0	0	0	eMIOS_0 channel 16
0	0	1	1	0	0	1	eMIOS_0 channel 17
0	0	1	1	0	1	0	eMIOS_0 channel 18
0	0	1	1	0	1	1	eMIOS_0 channel 19
0	0	1	1	1	0	0	eMIOS_0 channel 20
0	0	1	1	1	0	1	eMIOS_0 channel 21
0	0	1	1	1	1	0	eMIOS_0 channel 22
0	0	1	1	1	1	1	eMIOS_0 channel 23
0	1	x	x	x	x	x	eTPUA28
1	0	x	x	x	x	x	eMIOS_1 channel 6
1	1	x	x	x	x	x	ETRIG1 Pin

Table 8-1038. eQADC_A Command FIFO 4 Trigger Sources

cTSEL_4_A							eQADC_A Trigger Inputs
0	0	0	0	0	0	0	Not Connected (default)
0	0	0	0	0	0	1	RTI Trigger
0	0	0	0	0	1	0	PIT0 Trigger
0	0	0	0	0	1	1	PIT1 Trigger
0	0	0	0	1	0	0	PIT2 Trigger
0	0	0	0	1	0	1	PIT3 Trigger

Table continues on the next page...

Table 8-1038. eQADC_A Command FIFO 4 Trigger Sources (continued)

cTSEL_4_A							eQADC_A Trigger Inputs
0	0	0	0	1	1	0	eTPUA30
0	0	0	0	1	1	1	ETRIG1 pin
0	0	0	1	0	x	x	Reserved
0	0	0	1	1	0	0	eTPUA28
0	0	0	1	1	0	1	eTPUA29
0	0	0	1	1	1	0	eTPUA30
0	0	0	1	1	1	1	eTPUA31
0	0	1	0	0	0	0	eTPUB28
0	0	1	0	0	0	1	eTPUB29
0	0	1	0	0	1	0	eTPUB30
0	0	1	0	0	1	1	eTPUB31
0	0	1	0	1	0	0	eTPUC28
0	0	1	0	1	0	1	eTPUC29
0	0	1	0	1	1	0	eTPUC30
0	0	1	0	1	1	1	eTPUC31
0	0	1	1	0	0	0	eMIOS_0 channel 16
0	0	1	1	0	0	1	eMIOS_0 channel 17
0	0	1	1	0	1	0	eMIOS_0 channel 18
0	0	1	1	0	1	1	eMIOS_0 channel 19
0	0	1	1	1	0	0	eMIOS_0 channel 20
0	0	1	1	1	0	1	eMIOS_0 channel 21
0	0	1	1	1	1	0	eMIOS_0 channel 22
0	0	1	1	1	1	1	eMIOS_0 channel 23
0	1	x	x	x	x	x	eTPUA27
1	0	x	x	x	x	x	eMIOS_1 channel 5
1	1	x	x	x	x	x	ETRIG0 Pin

Table 8-1039. eQADC_A Command FIFO 5 Trigger Sources

cTSEL_5_A							eQADC_A Trigger Inputs
0	0	0	0	0	0	0	Not Connected (default)
0	0	0	0	0	0	1	RTI Trigger
0	0	0	0	0	1	0	PIT0 Trigger
0	0	0	0	0	1	1	PIT1 Trigger
0	0	0	0	1	0	0	PIT2 Trigger
0	0	0	0	1	0	1	PIT3 Trigger
0	0	0	0	1	1	0	Reserved
0	0	0	0	1	1	1	ETRIG0 pin
0	0	0	1	0	x	x	Reserved

Table continues on the next page...

Table 8-1039. eQADC_A Command FIFO 5 Trigger Sources (continued)

cTSEL_5_A							eQADC_A Trigger Inputs
0	0	0	1	1	0	0	eTPUA28
0	0	0	1	1	0	1	eTPUA29
0	0	0	1	1	1	0	eTPUA30
0	0	0	1	1	1	1	eTPUA31
0	0	1	0	0	0	0	eTPUB28
0	0	1	0	0	0	1	eTPUB29
0	0	1	0	0	1	0	eTPUB30
0	0	1	0	0	1	1	eTPUB31
0	0	1	0	1	0	0	eTPUC28
0	0	1	0	1	0	1	eTPUC29
0	0	1	0	1	1	0	eTPUC30
0	0	1	0	1	1	1	eTPUC31
0	0	1	1	0	0	0	eMIOS_0 channel 16
0	0	1	1	0	0	1	eMIOS_0 channel 17
0	0	1	1	0	1	0	eMIOS_0 channel 18
0	0	1	1	0	1	1	eMIOS_0 channel 19
0	0	1	1	1	0	0	eMIOS_0 channel 20
0	0	1	1	1	0	1	eMIOS_0 channel 21
0	0	1	1	1	1	0	eMIOS_0 channel 22
0	0	1	1	1	1	1	eMIOS_0 channel 23
0	1	x	x	x	x	x	eTPUA26
1	0	x	x	x	x	x	eMIOS_1 channel 4
1	1	x	x	x	x	x	ETRIG1 Pin

Address: 0h base + 910h offset = 910h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0								0							
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0								0							
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

SIU_ISEL4 field descriptions

Field	Description
0 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
1–7 CTSEL5_A	Trigger source selection for EQADC_A Command FIFO 5.

Table continues on the next page...

SIU_ISEL4 field descriptions (continued)

Field	Description
8 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
9–15 CTSEL4_A	Trigger source selection for EQADC_A Command FIFO 4.
16 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
17–23 CTSEL3_A	Trigger source selection for EQADC_A Command FIFO 3.
24 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
25–31 CTSEL2_A	Trigger source selection for EQADC_A Command FIFO2.

8.2.19 eQADC Command FIFO Trigger Source Registers (SIU_ISEL5)

The eQADC Command FIFO[1:0] Trigger Source Register (SIU_ISEL5) is used to select a trigger source for a command FIFO. SIU_ISEL5 is used to select a trigger source for command FIFO[1:0] in eQADC_A. The cTSEL (combined Trigger Select) field is used to configure one of many possible trigger sources for each command FIFO.

To trigger the eQADC, the trigger source must change to the state that the input to the command FIFO has been programmed to recognize. A command FIFO trigger input can be programmed to recognize either rising or falling edges, and low or high gated trigger types.

Table 8-1041. eQADC_A Command FIFO 0 Trigger Sources

cTSEL_0_A							eQADC_A Trigger Inputs
0	0	0	0	0	0	0	Not connected (default)
0	0	0	0	0	0	1	RTI Trigger
0	0	0	0	0	1	0	PIT0 Trigger
0	0	0	0	0	1	1	PIT1 Trigger
0	0	0	0	1	0	0	PIT2 Trigger
0	0	0	0	1	0	1	PIT3 Trigger
0	0	0	0	1	1	0	Reserved
0	0	0	0	1	1	1	ETRIG1 pin
0	0	0	1	0	x	x	Reserved
0	0	0	1	1	0	0	eTPUA28
0	0	0	1	1	0	1	eTPUA29
0	0	0	1	1	1	0	eTPUA30
0	0	0	1	1	1	1	eTPUA31

Table continues on the next page...

Table 8-1041. eQADC_A Command FIFO 0 Trigger Sources (continued)

cTSEL_0_A							eQADC_A Trigger Inputs
0	0	1	0	0	0	0	eTPUB28
0	0	1	0	0	0	1	eTPUB29
0	0	1	0	0	1	0	eTPUB30
0	0	1	0	0	1	1	eTPUB31
0	0	1	0	1	0	0	eTPUC28
0	0	1	0	1	0	1	eTPUC29
0	0	1	0	1	1	0	eTPUC30
0	0	1	0	1	1	1	eTPUC31
0	0	1	1	0	0	0	eMIOS_0 channel 16
0	0	1	1	0	0	1	eMIOS_0 channel 17
0	0	1	1	0	1	0	eMIOS_0 channel 18
0	0	1	1	0	1	1	eMIOS_0 channel 19
0	0	1	1	1	0	0	eMIOS_0 channel 20
0	0	1	1	1	0	1	eMIOS_0 channel 21
0	0	1	1	1	1	0	eMIOS_0 channel 22
0	0	1	1	1	1	1	eMIOS_0 channel 23
0	1	x	x	x	x	x	eTPUA30
1	0	x	x	x	x	x	eMIOS_1 channel 2
1	1	x	x	x	x	x	ETRIG0 Pin

Table 8-1042. eQADC_A Command FIFO 1 Trigger Sources

cTSEL_1_A							eQADC_A Trigger Inputs
0	0	0	0	0	0	0	Not Connected (default)
0	0	0	0	0	0	1	RTI Trigger
0	0	0	0	0	1	0	PIT0 Trigger
0	0	0	0	0	1	1	PIT1 Trigger
0	0	0	0	1	0	0	PIT2 Trigger
0	0	0	0	1	0	1	PIT3 Trigger
0	0	0	0	1	1	0	eTPUA7
0	0	0	0	1	1	1	ETRIG0 pin
0	0	0	1	0	x	x	Reserved
0	0	0	1	1	0	0	eTPUA28
0	0	0	1	1	0	1	eTPUA29
0	0	0	1	1	1	0	eTPUA30
0	0	0	1	1	1	1	eTPUA31
0	0	1	0	0	0	0	eTPUB28
0	0	1	0	0	0	1	eTPUB29
0	0	1	0	0	1	0	eTPUB30

Table continues on the next page...

Table 8-1042. eQADC_A Command FIFO 1 Trigger Sources (continued)

cTSEL_1_A							eQADC_A Trigger Inputs
0	0	1	0	0	1	1	eTPUB31
0	0	1	0	1	0	0	eTPUC28
0	0	1	0	1	0	1	eTPUC29
0	0	1	0	1	1	0	eTPUC30
0	0	1	0	1	1	1	eTPUC31
0	0	1	1	0	0	0	eMIOS_0 channel 16
0	0	1	1	0	0	1	eMIOS_0 channel 17
0	0	1	1	0	1	0	eMIOS_0 channel 18
0	0	1	1	0	1	1	eMIOS_0 channel 19
0	0	1	1	1	0	0	eMIOS_0 channel 20
0	0	1	1	1	0	1	eMIOS_0 channel 21
0	0	1	1	1	1	0	eMIOS_0 channel 22
0	0	1	1	1	1	1	eMIOS_0 channel 23
0	1	x	x	x	x	x	eTPUA31
1	0	x	x	x	x	x	eMIOS_1 channel 3
1	1	x	x	x	x	x	ETRIG1 Pin

Address: 0h base + 914h offset = 914h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0								0							
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R									0							
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

SIU_ISEL5 field descriptions

Field	Description
0 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
1–7 CTSEL1_A	Trigger source selection for EQADC_A Command FIFO 1.
8 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
9–15 CTSEL0_A	Trigger source selection for EQADC_A Command FIFO 0.
16–31 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

8.2.20 eQADC Command FIFO Trigger Source Registers (SIU_ISEL6)

The eQADC Command FIFO[5:2] Trigger Source Register (SIU_ISEL6) is used to select a trigger source for a command FIFO. SIU_ISEL6 is used to select a trigger source for command FIFO[5:2] in eQADC_B. The cTSEL (combined Trigger Select) field is used to configure one of many possible trigger sources for each command FIFO.

To trigger the eQADC, the trigger source must change to the state that the input to the command FIFO has been programmed to recognize. A command FIFO trigger input can be programmed to recognize either rising or falling edges, and low or high gated trigger types.

Table 8-1044. eQADC_B Command FIFO 2 Trigger Sources

cTSEL_2_B							eQADC_B Trigger Inputs
0	0	0	0	0	0	0	Not Connected (default)
0	0	0	0	0	0	1	RTI Trigger
0	0	0	0	0	1	0	PIT0 Trigger
0	0	0	0	0	1	1	PIT1 Trigger
0	0	0	0	1	0	0	PIT2 Trigger
0	0	0	0	1	0	1	PIT3 Trigger
0	0	0	0	1	1	0	eTPUA14
0	0	0	0	1	1	1	ETRIG1 pin
0	0	0	1	0	x	x	Reserved
0	0	0	1	1	0	0	eTPUA28
0	0	0	1	1	0	1	eTPUA29
0	0	0	1	1	1	0	eTPUA30
0	0	0	1	1	1	1	eTPUA31
0	0	1	0	0	0	0	eTPUB28
0	0	1	0	0	0	1	eTPUB29
0	0	1	0	0	1	0	eTPUB30
0	0	1	0	0	1	1	eTPUB31
0	0	1	0	1	0	0	eTPUC28
0	0	1	0	1	0	1	eTPUC29
0	0	1	0	1	1	0	eTPUC30
0	0	1	0	1	1	1	eTPUC31
0	0	1	1	0	0	0	eMIOS_0 channel 16
0	0	1	1	0	0	1	eMIOS_0 channel 17
0	0	1	1	0	1	0	eMIOS_0 channel 18
0	0	1	1	0	1	1	eMIOS_0 channel 19
0	0	1	1	1	0	0	eMIOS_0 channel 20
0	0	1	1	1	0	1	eMIOS_0 channel 21

Table continues on the next page...

Table 8-1044. eQADC_B Command FIFO 2 Trigger Sources (continued)

cTSEL_2_B							eQADC_B Trigger Inputs
0	0	1	1	1	1	0	eMIOS_0 channel 22
0	0	1	1	1	1	1	eMIOS_0 channel 23
0	1	x	x	x	x	x	eTPUA29
1	0	x	x	x	x	x	eMIOS_1 channel 7
1	1	x	x	x	x	x	ETRIG0 Pin

Table 8-1045. eQADC_B Command FIFO 3 Trigger Sources

cTSEL_3_B							eQADC_B Trigger Inputs
0	0	0	0	0	0	0	Not Connected (default)
0	0	0	0	0	0	1	RTI Trigger
0	0	0	0	0	1	0	PIT0 Trigger
0	0	0	0	0	1	1	PIT1 Trigger
0	0	0	0	1	0	0	PIT2 Trigger
0	0	0	0	1	0	1	PIT3 Trigger
0	0	0	0	1	1	0	eTPUA22
0	0	0	0	1	1	1	ETRIG0 pin
0	0	0	1	0	x	x	Reserved
0	0	0	1	1	0	0	eTPUA28
0	0	0	1	1	0	1	eTPUA29
0	0	0	1	1	1	0	eTPUA30
0	0	0	1	1	1	1	eTPUA31
0	0	1	0	0	0	0	eTPUB28
0	0	1	0	0	0	1	eTPUB29
0	0	1	0	0	1	0	eTPUB30
0	0	1	0	0	1	1	eTPUB31
0	0	1	0	1	0	0	eTPUC28
0	0	1	0	1	0	1	eTPUC29
0	0	1	0	1	1	0	eTPUC30
0	0	1	0	1	1	1	eTPUC31
0	0	1	1	0	0	0	eMIOS_0 channel 16
0	0	1	1	0	0	1	eMIOS_0 channel 17
0	0	1	1	0	1	0	eMIOS_0 channel 18
0	0	1	1	0	1	1	eMIOS_0 channel 19
0	0	1	1	1	0	0	eMIOS_0 channel 20
0	0	1	1	1	0	1	eMIOS_0 channel 21
0	0	1	1	1	1	0	eMIOS_0 channel 22
0	0	1	1	1	1	1	eMIOS_0 channel 23
0	1	x	x	x	x	x	eTPUA28

Table continues on the next page...

Table 8-1045. eQADC_B Command FIFO 3 Trigger Sources (continued)

cTSEL_3_B							eQADC_B Trigger Inputs
1	0	x	x	x	x	x	eMIOS_1 channel 6
1	1	x	x	x	x	x	ETRIG1 Pin

Table 8-1046. eQADC_B Command FIFO 4 Trigger Sources

cTSEL_4_B							eQADC_B Trigger Inputs
0	0	0	0	0	0	0	Not Connected (default)
0	0	0	0	0	0	1	RTI Trigger
0	0	0	0	0	1	0	PIT0 Trigger
0	0	0	0	0	1	1	PIT1 Trigger
0	0	0	0	1	0	0	PIT2 Trigger
0	0	0	0	1	0	1	PIT3 Trigger
0	0	0	0	1	1	0	eTPUA30
0	0	0	0	1	1	1	ETRIG1 pin
0	0	0	1	0	x	x	Reserved
0	0	0	1	1	0	0	eTPUA28
0	0	0	1	1	0	1	eTPUA29
0	0	0	1	1	1	0	eTPUA30
0	0	0	1	1	1	1	eTPUA31
0	0	1	0	0	0	0	eTPUB28
0	0	1	0	0	0	1	eTPUB29
0	0	1	0	0	1	0	eTPUB30
0	0	1	0	0	1	1	eTPUB31
0	0	1	0	1	0	0	eTPUC28
0	0	1	0	1	0	1	eTPUC29
0	0	1	0	1	1	0	eTPUC30
0	0	1	0	1	1	1	eTPUC31
0	0	1	1	0	0	0	eMIOS_0 channel 16
0	0	1	1	0	0	1	eMIOS_0 channel 17
0	0	1	1	0	1	0	eMIOS_0 channel 18
0	0	1	1	0	1	1	eMIOS_0 channel 19
0	0	1	1	1	0	0	eMIOS_0 channel 20
0	0	1	1	1	0	1	eMIOS_0 channel 21
0	0	1	1	1	1	0	eMIOS_0 channel 22
0	0	1	1	1	1	1	eMIOS_0 channel 23
0	1	x	x	x	x	x	eTPUA27
1	0	x	x	x	x	x	eMIOS_1 channel 5
1	1	x	x	x	x	x	ETRIG0 Pin

Table 8-1047. eQADC_B Command FIFO 5 Trigger Sources

cTSEL_5_B							eQADC_B Trigger Inputs
0	0	0	0	0	0	0	Not Connected (default)
0	0	0	0	0	0	1	RTI Trigger
0	0	0	0	0	1	0	PIT0 Trigger
0	0	0	0	0	1	1	PIT1 Trigger
0	0	0	0	1	0	0	PIT2 Trigger
0	0	0	0	1	0	1	PIT3 Trigger
0	0	0	0	1	1	0	Reserved
0	0	0	0	1	1	1	ETRIG0 pin
0	0	0	1	0	x	x	Reserved
0	0	0	1	1	0	0	eTPUA28
0	0	0	1	1	0	1	eTPUA29
0	0	0	1	1	1	0	eTPUA30
0	0	0	1	1	1	1	eTPUA31
0	0	1	0	0	0	0	eTPUB28
0	0	1	0	0	0	1	eTPUB29
0	0	1	0	0	1	0	eTPUB30
0	0	1	0	0	1	1	eTPUB31
0	0	1	0	1	0	0	eTPUC28
0	0	1	0	1	0	1	eTPUC29
0	0	1	0	1	1	0	eTPUC30
0	0	1	0	1	1	1	eTPUC31
0	0	1	1	0	0	0	eMIOS_0 channel 16
0	0	1	1	0	0	1	eMIOS_0 channel 17
0	0	1	1	0	1	0	eMIOS_0 channel 18
0	0	1	1	0	1	1	eMIOS_0 channel 19
0	0	1	1	1	0	0	eMIOS_0 channel 20
0	0	1	1	1	0	1	eMIOS_0 channel 21
0	0	1	1	1	1	0	eMIOS_0 channel 22
0	0	1	1	1	1	1	eMIOS_0 channel 23
0	1	x	x	x	x	x	eTPUA26
1	0	x	x	x	x	x	eMIOS_1 channel 4
1	1	x	x	x	x	x	ETRIG1 Pin

Address: 0h base + 918h offset = 918h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0								0							
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	CTSEL3_B							0	CTSEL2_B						
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

SIU_ISEL6 field descriptions

Field	Description
0 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
1–7 CTSEL5_B	Trigger source selection for EQADC_B Command FIFO 5.
8 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
9–15 CTSEL4_B	Trigger source selection for EQADC_B Command FIFO 4.
16 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
17–23 CTSEL3_B	Trigger source selection for EQADC_B Command FIFO 3.
24 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
25–31 CTSEL2_B	Trigger source selection for EQADC_B Command FIFO 2.

8.2.21 eQADC Command FIFO Trigger Source Registers (SIU_ISEL7)

The eQADC Command FIFO[1:0] Trigger Source Register (SIU_ISEL7) is used to select a trigger source for a command FIFO. SIU_ISEL7 is used to select a trigger source for command FIFO[1:0] in eQADC_B. The cTSEL (combined Trigger Select) field is used to configure one of many possible trigger sources for each command FIFO.

To trigger the eQADC, the trigger source must change to the state that the input to the command FIFO has been programmed to recognize. A command FIFO trigger input can be programmed to recognize either rising or falling edges, and low or high gated trigger types.

Table 8-1049. eQADC_B Command FIFO 0 Trigger Sources

cTSEL_0_B							eQADC_B Trigger Inputs
0	0	0	0	0	0	0	Not Connected (default)
0	0	0	0	0	0	1	RTI Trigger
0	0	0	0	0	1	0	PIT0 Trigger
0	0	0	0	0	1	1	PIT1 Trigger
0	0	0	0	1	0	0	PIT2 Trigger

Table continues on the next page...

Table 8-1049. eQADC_B Command FIFO 0 Trigger Sources (continued)

cTSEL_0_B							eQADC_B Trigger Inputs
0	0	0	0	1	0	1	PIT3 Trigger
0	0	0	0	1	1	0	Reserved
0	0	0	0	1	1	1	ETRIG1 pin
0	0	0	1	0	x	x	Reserved
0	0	0	1	1	0	0	eTPUA28
0	0	0	1	1	0	1	eTPUA29
0	0	0	1	1	1	0	eTPUA30
0	0	0	1	1	1	1	eTPUA31
0	0	1	0	0	0	0	eTPUB28
0	0	1	0	0	0	1	eTPUB29
0	0	1	0	0	1	0	eTPUB30
0	0	1	0	0	1	1	eTPUB31
0	0	1	0	1	0	0	eTPUC28
0	0	1	0	1	0	1	eTPUC29
0	0	1	0	1	1	0	eTPUC30
0	0	1	0	1	1	1	eTPUC31
0	0	1	1	0	0	0	eMIOS_0 channel 16
0	0	1	1	0	0	1	eMIOS_0 channel 17
0	0	1	1	0	1	0	eMIOS_0 channel 18
0	0	1	1	0	1	1	eMIOS_0 channel 19
0	0	1	1	1	0	0	eMIOS_0 channel 20
0	0	1	1	1	0	1	eMIOS_0 channel 21
0	0	1	1	1	1	0	eMIOS_0 channel 22
0	0	1	1	1	1	1	eMIOS_0 channel 23
0	1	x	x	x	x	x	eTPUA30
1	0	x	x	x	x	x	eMIOS_1 channel 2
1	1	x	x	x	x	x	ETRIG0 Pin

Table 8-1050. eQADC_B Command FIFO 1 Trigger Sources

cTSEL_1_B							eQADC_B Trigger Inputs
0	0	0	0	0	0	0	Not Connected (default)
0	0	0	0	0	0	1	RTI Trigger
0	0	0	0	0	1	0	PIT0 Trigger
0	0	0	0	0	1	1	PIT1 Trigger
0	0	0	0	1	0	0	PIT2 Trigger
0	0	0	0	1	0	1	PIT3 Trigger
0	0	0	0	1	1	0	eTPUA7
0	0	0	0	1	1	1	ETRIG0 pin

Table continues on the next page...

Table 8-1050. eQADC_B Command FIFO 1 Trigger Sources (continued)

cTSEL_1_B							eQADC_B Trigger Inputs
0	0	0	1	0	x	x	Reserved
0	0	0	1	1	0	0	eTPUA28
0	0	0	1	1	0	1	eTPUA29
0	0	0	1	1	1	0	eTPUA30
0	0	0	1	1	1	1	eTPUA31
0	0	1	0	0	0	0	eTPUB28
0	0	1	0	0	0	1	eTPUB29
0	0	1	0	0	1	0	eTPUB30
0	0	1	0	0	1	1	eTPUB31
0	0	1	0	1	0	0	eTPUC28
0	0	1	0	1	0	1	eTPUC29
0	0	1	0	1	1	0	eTPUC30
0	0	1	0	1	1	1	eTPUC31
0	0	1	1	0	0	0	eMIOS_0 channel 16
0	0	1	1	0	0	1	eMIOS_0 channel 17
0	0	1	1	0	1	0	eMIOS_0 channel 18
0	0	1	1	0	1	1	eMIOS_0 channel 19
0	0	1	1	1	0	0	eMIOS_0 channel 20
0	0	1	1	1	0	1	eMIOS_0 channel 21
0	0	1	1	1	1	0	eMIOS_0 channel 22
0	0	1	1	1	1	1	eMIOS_0 channel 23
0	1	x	x	x	x	x	eTPUA31
1	0	x	x	x	x	x	eMIOS_1 channel 3
1	1	x	x	x	x	x	ETRIG1 Pin

Address: 0h base + 91Ch offset = 91Ch

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0								0							
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R									0							
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

SIU_ISEL7 field descriptions

Field	Description
0 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

Table continues on the next page...

SIU_ISEL7 field descriptions (continued)

Field	Description
1–7 CTSEL1_B	Trigger source selection for EQADC_B Command FIFO 1.
8 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
9–15 CTSEL0_B	Trigger source selection for EQADC_B Command FIFO 0.
16–31 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

8.2.22 eTPU_A Input Select Register (SIU_ISEL8)

The SIU_ISEL8 register is used to multiplex the eTPU_A[24:29] inputs. These 6 eTPU_A channels can come from the output of the DSPI or the corresponding pad. When SIU_ISEL8 is in its default state, the eTPU pins listed in the figure below will not be enabled as inputs, irrespective of the SIU_PCR[PA] field (PCR138-143).

Address: 0h base + 920h offset = 920h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0										ETPU29		0		ETPU28	
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0		ETPU27		0		ETPU26		0		ETPU25		0		ETPU24	
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

SIU_ISEL8 field descriptions

Field	Description
0–10 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
11 ETPU29	eTPU29 input select. Specifies the source of the eTPU29 channel input. 0 DSPI_B Serialized input 8. 1 eTPUA29 pin or ANB10 pin, as selected by SIU_IMUX7[MUXSEL13].

Table continues on the next page...

SIU_ISEL8 field descriptions (continued)

Field	Description
12–14 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
15 ETPU28	eTPU28 input select. Specifies the source of the eTPU28 channel input. 0 DSPI_B Serialized input 9. 1 eTPUA28 pin or ANB11 pin, as selected by SIU_IMUX7[MUXSEL12].
16–18 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
19 ETPU27	eTPU27 input select. Specifies the source of the eTPU27 channel input. 0 DSPI_B Serialized input 10. 1 eTPUA27 pin or ANB12 pin, as selected by SIU_IMUX7[MUXSEL11].
20–22 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
23 ETPU26	eTPU26 input select. Specifies the source of the eTPU26 channel input. 0 DSPI_B Serialized input 11. 1 eTPUA26 pin or ANB13 pin, as selected by SIU_IMUX7[MUXSEL10].
24–26 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
27 ETPU25	eTPU25 input select. Specifies the source of the eTPU25 channel input. 0 DSPI_B Serialized input 12. 1 eTPUA25 pin or ANB14 pin, as selected by SIU_IMUX7[MUXSEL9].
28–30 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
31 ETPU24	eTPU24 input select. Specifies the source of the eTPU24 channel input. 0 DSPI_B Serialized input 13. 1 eTPUA24 pin or ANB15 pin, as selected by SIU_IMUX7[MUXSEL8].

8.2.23 eQADC Advance Trigger Source Register (SIU_ISEL9)

The eQADC streaming mode requires a second trigger for command queue 0. Selectable source options for this trigger consist of eTPU, eMIOS or PIT channels. This mux register selects the source of the command queue 0 trigger.

Table 8-1053. eQADC_A CFIFO0 Advance Trigger Source Selection

eETSEL0ADV_A					eQADC Trigger Source
0	0	0	0	0	Reserved
0	0	0	0	1	RTI Trigger
0	0	0	1	0	PIT0 Trigger
0	0	0	1	1	PIT1 Trigger

Table continues on the next page...

Table 8-1053. eQADC_A CFIFO0 Advance Trigger Source Selection (continued)

eETSEL0ADV_A					eQADC Trigger Source
0	0	1	0	0	PIT2 Trigger
0	0	1	0	1	PIT3 Trigger
0	0	1	1	0	Reserved
0	0	1	1	1	Reserved
0	1	0	0	0	eTPUA30 AND PIT0
0	1	0	0	1	eTPUA30 AND PIT1
0	1	0	1	0	Reserved
0	1	0	1	1	Reserved
0	1	1	0	0	eTPUA28
0	1	1	0	1	eTPUA29
0	1	1	1	0	eTPUA30
0	1	1	1	1	eTPUA31
1	0	0	0	0	eTPUC28
1	0	0	0	1	eTPUC29
1	0	0	1	0	eTPUC30
1	0	0	1	1	eTPUC31
1	0	1	0	0	eMIOS_1 channel 2 AND PIT2
1	0	1	0	1	eMIOS_1 channel 2 AND PIT3
1	0	1	1	0	Reserved
1	0	1	1	1	Reserved
1	1	0	0	0	Reserved
1	1	0	0	1	Reserved
1	1	0	1	0	Reserved
1	1	0	1	1	Reserved
1	1	1	0	0	Reserved
1	1	1	0	1	Reserved
1	1	1	1	0	Reserved
1	1	1	1	1	eMIOS_0 channel 23

Table 8-1054. eQADC_B CFIFO0 Advance Trigger Source Selection

eETSEL0ADV_B					eQADC Trigger Source
0	0	0	0	0	Reserved
0	0	0	0	1	RTI Trigger
0	0	0	1	0	PIT0 Trigger
0	0	0	1	1	PIT1 Trigger
0	0	1	0	0	PIT2 Trigger
0	0	1	0	1	PIT3 Trigger

Table continues on the next page...

**Table 8-1054. eQADC_B CFIFO0 Advance Trigger Source Selection
(continued)**

eETSEL0ADV_B					eQADC Trigger Source
0	0	1	1	0	Reserved
0	0	1	1	1	Reserved
0	1	0	0	0	eTPUA30 AND PIT0
0	1	0	0	1	eTPUA30 AND PIT1
0	1	0	1	0	Reserved
0	1	0	1	1	Reserved
0	1	1	0	0	eTPUA28
0	1	1	0	1	eTPUA29
0	1	1	1	0	eTPUA30
0	1	1	1	1	eTPUA31
1	0	0	0	0	eTPUC28
1	0	0	0	1	eTPUC29
1	0	0	1	0	eTPUC30
1	0	0	1	1	eTPUC31
1	0	1	0	0	eMIOS_1 channel 2 AND PIT2
1	0	1	0	1	eMIOS_1 channel 2 AND PIT3
1	0	1	1	0	Reserved
1	0	1	1	1	Reserved
1	1	0	0	0	Reserved
1	1	0	0	1	Reserved
1	1	0	1	0	Reserved
1	1	0	1	1	Reserved
1	1	1	0	0	Reserved
1	1	1	0	1	Reserved
1	1	1	1	0	Reserved
1	1	1	1	1	eMIOS_0 channel 23

Address: 0h base + 924h offset = 924h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0										EETSEL0ADV_A					0										EETSEL0ADV_B						
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

SIU_ISEL9 field descriptions

Field	Description
0–10 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
11–15 EETSEL0ADV_A	EQADC_A CFIFO0 Advance Trigger Source Selection

Table continues on the next page...

SIU_ISEL9 field descriptions (continued)

Field	Description
16–26 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
27–31 EETSEL0ADV_B	EQADC_B CFIFO0 Advance Trigger Source Selection

8.2.24 Decimation Filter Register 1 (SIU_DECFIL1)

This register selects the source of the control signals for the integrators of decimation filters A to D. Each ZSELx field of this register may be programmed to route a single eTPU channel simultaneously to the Zero, Integrate and Read control inputs of one integrator. Each HSELx field may be programmed to route a single eTPU channel to the Halt control input of one integrator. See the Decimation Filter Chapter for details on the Zero, Integrate and Halt configuration and functionality.

Address: 0h base + 928h offset = 928h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R																																
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

SIU_DECFIL1 field descriptions

Field	Description
0–3 HSEL0	Halt Input Select for Decimation Filter D. 0000 Unused. 0001 eTPUB Channel 22. 0010 eTPUB Channel 23. 0011 eTPUB Channel 24. 0100 eTPUB Channel 25. 0101 eTPUC Channel 22. 0110 eTPUC Channel 23. 0111 eTPUC Channel 24. 1000 eTPUC Channel 25.
4–7 ZSEL0	ZIR Input Select for Decimation Filter D. 0000 Unused. 0001 eTPUB Channel 22. 0010 eTPUB Channel 23. 0011 eTPUB Channel 24. 0100 eTPUB Channel 25. 0101 eTPUC Channel 22.

Table continues on the next page...

SIU_DECFIL1 field descriptions (continued)

Field	Description
	0110 eTPUC Channel 23. 0111 eTPUC Channel 24. 1000 eTPUC Channel 25.
8–11 HSELC	Halt Input Select for Decimation Filter C. 0000 Unused 0001 eTPUB Channel 22. 0010 eTPUB Channel 23. 0011 eTPUB Channel 24. 0100 eTPUB Channel 25. 0101 eTPUC Channel 22. 0110 eTPUC Channel 23. 0111 eTPUC Channel 24. 1000 eTPUC Channel 25.
12–15 ZSELC	ZIR Input Select for Decimation Filter C. 0000 Unused. 0001 eTPUB Channel 22. 0010 eTPUB Channel 23. 0011 eTPUB Channel 24. 0100 eTPUB Channel 25. 0101 eTPUC Channel 22. 0110 eTPUC Channel 23. 0111 eTPUC Channel 24. 1000 eTPUC Channel 25.
16–19 HSELB	Halt Input Select for Decimation Filter B. 0000 Unused. 0001 eTPUA Channel 22. 0010 eTPUA Channel 23. 0011 eTPUA Channel 24. 0100 eTPUA Channel 25. 0101 eTPUC Channel 22. 0110 eTPUC Channel 23. 0111 eTPUC Channel 24. 1000 eTPUC Channel 25.
20–23 ZSELB	ZIR Input Select for Decimation Filter B. 0000 Unused. 0001 eTPUA Channel 22. 0010 eTPUA Channel 23. 0011 eTPUA Channel 24. 0100 eTPUA Channel 25. 0101 eTPUC Channel 22. 0110 eTPUC Channel 23. 0111 eTPUC Channel 24. 1000 eTPUC Channel 25.

Table continues on the next page...

SIU_DECFIL1 field descriptions (continued)

Field	Description
24–27 HSELA	Halt Input Select for Decimation Filter A. 0000 Unused. 0001 eTPUA Channel 22. 0010 eTPUA Channel 23. 0011 eTPUA Channel 24. 0100 eTPUA Channel 25. 0101 eTPUC Channel 22. 0110 eTPUC Channel 23. 0111 eTPUC Channel 24. 1000 eTPUC Channel 25.
28–31 ZSELA	ZIR Input Select for Decimation Filter A. 0000 Unused. 0001 eTPUA Channel 22. 0010 eTPUA Channel 23. 0011 eTPUA Channel 24. 0100 eTPUA Channel 25. 0101 eTPUC Channel 22. 0110 eTPUC Channel 23. 0111 eTPUC Channel 24. 1000 eTPUC Channel 25.

8.2.25 Decimation Filter Register 2 (SIU_DECFIL2)

This register selects the source of the control signals for the integrators of decimation filters E to H. Each ZSELx field of this register may be programmed to route a single eTPU channel simultaneously to the Zero, Integrate and Read control inputs of one integrator. Each HSELx field may be programmed to route a single eTPU channel to the Halt control input of one integrator. See the Decimation Filter chapter for details on the Zero, Integrate, Read and Halt control configuration and functionality.

Address: 0h base + 92Ch offset = 92Ch

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R																																
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

SIU_DECFIL2 field descriptions

Field	Description
0–3 HSELH	Halt Input Select for Decimation Filter H.

Table continues on the next page...

SIU_DECFIL2 field descriptions (continued)

Field	Description
	0000 Unused. 0001 eTPUB Channel 18. 0010 eTPUB Channel 19. 0011 eTPUB Channel 20. 0100 eTPUB Channel 21. 0101 eTPUC Channel 18. 0110 eTPUC Channel 19. 0111 eTPUC Channel 20. 1000 eTPUC Channel 21.
4–7 ZSELH	ZIR Input Select for Decimation Filter H. 0000 Unused. 0001 eTPUB Channel 18. 0010 eTPUB Channel 19. 0011 eTPUB Channel 20. 0100 eTPUB Channel 21. 0101 eTPUC Channel 18. 0110 eTPUC Channel 19. 0111 eTPUC Channel 20. 1000 eTPUC Channel 21.
8–11 HSELG	Halt Input Select for Decimation Filter G. 0000 Unused 0001 eTPUB Channel 18. 0010 eTPUB Channel 19. 0011 eTPUB Channel 20. 0100 eTPUB Channel 21. 0101 eTPUC Channel 18. 0110 eTPUC Channel 19. 0111 eTPUC Channel 20. 1000 eTPUC Channel 21.
12–15 ZSELG	ZIR Input Select for Decimation Filter G. 0000 Unused. 0001 eTPUB Channel 18. 0010 eTPUB Channel 19. 0011 eTPUB Channel 20. 0100 eTPUB Channel 21. 0101 eTPUC Channel 18. 0110 eTPUC Channel 19. 0111 eTPUC Channel 20. 1000 eTPUC Channel 21.
16–19 HSELF	Halt Input Select for Decimation Filter F. 0000 Unused. 0001 eTPUA Channel 18. 0010 eTPUA Channel 19. 0011 eTPUA Channel 20.

Table continues on the next page...

SIU_DECFIL2 field descriptions (continued)

Field	Description
	0100 eTPUA Channel 21. 0101 eTPUC Channel 18. 0110 eTPUC Channel 19. 0111 eTPUC Channel 20. 1000 eTPUC Channel 21.
20–23 ZSELF	ZIR Input Select for Decimation Filter F. 0000 Unused. 0001 eTPUA Channel 18. 0010 eTPUA Channel 19. 0011 eTPUA Channel 20. 0100 eTPUA Channel 21. 0101 eTPUC Channel 18. 0110 eTPUC Channel 19. 0111 eTPUC Channel 20. 1000 eTPUC Channel 21.
24–27 HSELE	Halt Input Select for Decimation Filter E. 0000 Unused. 0001 eTPUA Channel 18. 0010 eTPUA Channel 19. 0011 eTPUA Channel 20. 0100 eTPUA Channel 21. 0101 eTPUC Channel 18. 0110 eTPUC Channel 19. 0111 eTPUC Channel 20. 1000 eTPUC Channel 21.
28–31 ZSELE	ZIR Input Select for Decimation Filter E. 0000 Unused. 0001 eTPUA Channel 18. 0010 eTPUA Channel 19. 0011 eTPUA Channel 20. 0100 eTPUA Channel 21. 0101 eTPUC Channel 18. 0110 eTPUC Channel 19. 0111 eTPUC Channel 20. 1000 eTPUC Channel 21.

8.2.26 Decimation Filter Register 3 (SIU_DECFIL3)

This register selects the source of the control signals for the integrators of decimation filters I to L. Each ZSELx field of this register may be programmed to route a single eTPU channel simultaneously to the Zero, Integrate and Read control inputs of one integrator. Each HSELx field may be programmed to route a single eTPU channel to the Halt control input of one integrator. See the Decimation Filter chapter for details on the Zero, Integrate, Read and Halt control configuration and functionality.

Address: 0h base + 930h offset = 930h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R																																
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

SIU_DECFIL3 field descriptions

Field	Description
0–3 HSELL	Halt Input Select for Decimation Filter L. 0000 Unused. 0001 eTPUB Channel 14. 0010 eTPUB Channel 15. 0011 eTPUB Channel 16. 0100 eTPUB Channel 17. 0101 eTPUC Channel 14. 0110 eTPUC Channel 15. 0111 eTPUC Channel 16. 1000 eTPUC Channel 17.
4–7 ZSELL	ZIR Input Select for Decimation Filter L. 0000 Unused. 0001 eTPUB Channel 14. 0010 eTPUB Channel 15. 0011 eTPUB Channel 16. 0100 eTPUB Channel 17. 0101 eTPUC Channel 14. 0110 eTPUC Channel 15. 0111 eTPUC Channel 16. 1000 eTPUC Channel 17.
8–11 HSELK	Halt Input Select for Decimation Filter K. 0000 Unused 0001 eTPUB Channel 14. 0010 eTPUB Channel 15. 0011 eTPUB Channel 16.

Table continues on the next page...

SIU_DECFIL3 field descriptions (continued)

Field	Description
	0100 eTPUB Channel 17. 0101 eTPUC Channel 14. 0110 eTPUC Channel 15. 0111 eTPUC Channel 16. 1000 eTPUC Channel 17.
12–15 ZSELK	ZIR Input Select for Decimation Filter K. 0000 Unused. 0001 eTPUB Channel 14. 0010 eTPUB Channel 15. 0011 eTPUB Channel 16. 0100 eTPUB Channel 17. 0101 eTPUC Channel 14. 0110 eTPUC Channel 15. 0111 eTPUC Channel 16. 1000 eTPUC Channel 17.
16–19 HSELJ	Halt Input Select for Decimation Filter J. 0000 Unused. 0001 eTPUA Channel 14. 0010 eTPUA Channel 15. 0011 eTPUA Channel 16. 0100 eTPUA Channel 17. 0101 eTPUC Channel 14. 0110 eTPUC Channel 15. 0111 eTPUC Channel 16. 1000 eTPUC Channel 17.
20–23 ZSELJ	ZIR Input Select for Decimation Filter J. 0000 Unused. 0001 eTPUA Channel 14. 0010 eTPUA Channel 15. 0011 eTPUA Channel 16. 0100 eTPUA Channel 17. 0101 eTPUC Channel 14. 0110 eTPUC Channel 15. 0111 eTPUC Channel 16. 1000 eTPUC Channel 17.
24–27 HSELI	Halt Input Select for Decimation Filter I. 0000 Unused. 0001 eTPUA Channel 14. 0010 eTPUA Channel 15. 0011 eTPUA Channel 16. 0100 eTPUA Channel 17. 0101 eTPUC Channel 14. 0110 eTPUC Channel 15.

Table continues on the next page...

SIU_DECFIL3 field descriptions (continued)

Field	Description
	0111 eTPUC Channel 16. 1000 eTPUC Channel 17.
28–31 ZSELI	ZIR Input Select for Decimation Filter I. 0000 Unused. 0001 eTPUA Channel 14. 0010 eTPUA Channel 15. 0011 eTPUA Channel 16. 0100 eTPUA Channel 17. 0101 eTPUC Channel 14. 0110 eTPUC Channel 15. 0111 eTPUC Channel 16. 1000 eTPUC Channel 17.

8.2.27 Decimation Filter Register 4 (SIU_DECFIL4)

The SIU_DECFIL4 register contains bit fields that specify which eTPU output is used to trigger the decimation filter result output buffer for decimation filters A to H.

Table 8-1059. Decimation Filter Triggered Output Source Selection for DECFIL4

Field Name	Code ¹				Source
TRIG_SRCA	0	0	0	0	Not connected (default)
	0	0	0	1	eTPUA22
	0	0	1	0	eTPUA23
	0	0	1	1	eTPUA24
	0	1	0	0	eTPUA25
TRIG_SRCB	0	0	0	0	Not connected (default)
	0	0	0	1	eTPUA22
	0	0	1	0	eTPUA23
	0	0	1	1	eTPUA24
	0	1	0	0	eTPUA25
TRIG_SRCC	0	0	0	0	Not connected (default)
	0	0	0	1	eTPUA22
	0	0	1	0	eTPUA23
	0	0	1	1	eTPUA24
	0	1	0	0	eTPUA25

Table continues on the next page...

Table 8-1059. Decimation Filter Triggered Output Source Selection for DECFIL4 (continued)

Field Name	Code ¹				Source
TRIG_SRCD	0	0	0	0	Not connected (default)
	0	0	0	1	eTPUA22
	0	0	1	0	eTPUA23
	0	0	1	1	eTPUA24
	0	1	0	0	eTPUA25
TRIG_SRCE	0	0	0	0	Not connected (default)
	0	0	0	1	eTPUA22
	0	0	1	0	eTPUA23
	0	0	1	1	eTPUA24
	0	1	0	0	eTPUA25
TRIG_SRCF	0	0	0	0	Not connected (default)
	0	0	0	1	eTPUA22
	0	0	1	0	eTPUA23
	0	0	1	1	eTPUA24
	0	1	0	0	eTPUA25
TRIG_SRCG	0	0	0	0	Not connected (default)
	0	0	0	1	eTPUA22
	0	0	1	0	eTPUA23
	0	0	1	1	eTPUA24
	0	1	0	0	eTPUA25
TRIG_SRCH	0	0	0	0	Not connected (default)
	0	0	0	1	eTPUA22
	0	0	1	0	eTPUA23
	0	0	1	1	eTPUA24
	0	1	0	0	eTPUA25

1. Selecting a code from 0101 to 1111 results in the same behavior as selecting 0000.

Address: 0h base + 934h offset = 934h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R																																
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

SIU_DECFIL4 field descriptions

Field	Description
0–3 TRIG_SRCH	See the table in this register's description.
4–7 TRIG_SRCG	See the table in this register's description.

Table continues on the next page...

SIU_DECFIL4 field descriptions (continued)

Field	Description
8–11 TRIG_SRCF	See the table in this register's description.
12–15 TRIG_SRCE	See the table in this register's description.
16–19 TRIG_SRCF	See the table in this register's description.
20–23 TRIG_SRCC	See the table in this register's description.
24–27 TRIG_SRCB	See the table in this register's description.
28–31 TRIG_SRCB	See the table in this register's description.

1. Selecting a code from 0101 to 1111 results in the same behavior as selecting 0000.

8.2.28 Decimation Filter Register 5 (SIU_DECFIL5)

The SIU_DECFIL5 register contains bit fields that specify which eTPU output is used to trigger the decimation filter result output buffer for decimation filters I to L.

Table 8-1061. Decimation Filter Triggered Output Source Selection for DECFIL5.

Field Name	Code ¹				Source
TRIG_SRCI	0	0	0	0	Not connected (default)
	0	0	0	1	eTPUA22
	0	0	1	0	eTPUA23
	0	0	1	1	eTPUA24
	0	1	0	0	eTPUA25
TRIG_SRCJ	0	0	0	0	Not connected (default)
	0	0	0	1	eTPUA22
	0	0	1	0	eTPUA23
	0	0	1	1	eTPUA24
	0	1	0	0	eTPUA25
TRIG_SRCB	0	0	0	0	Not connected (default)
	0	0	0	1	eTPUA22
	0	0	1	0	eTPUA23
	0	0	1	1	eTPUA24
	0	1	0	0	eTPUA25

Table continues on the next page...

Table 8-1061. Decimation Filter Triggered Output Source Selection for DECFIL5.
(continued)

Field Name	Code ¹				Source
TRIG_SRCL	0	0	0	0	Not connected (default)
	0	0	0	1	eTPUA22
	0	0	1	0	eTPUA23
	0	0	1	1	eTPUA24
	0	1	0	0	eTPUA25

1. Selecting a code from 0101 to 1111 results in the same behavior as selecting 0000.

Address: 0h base + 938h offset = 938h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	TRIG_SRCL				TRIG_SRCK				TRIG_SRCJ				TRIG_SRCI				0															
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	

SIU_DECFIL5 field descriptions

Field	Description
0–3 TRIG_SRCL	See the table in this register's description.
4–7 TRIG_SRCK	See the table in this register's description.
8–11 TRIG_SRCJ	See the table in this register's description.
12–15 TRIG_SRCI	See the table in this register's description.
16–31 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

1. Selecting a code from 0101 to 1111 results in the same behavior as selecting 0000.

8.2.29 Reaction Module Trigger Selection Register (SIU_REACTSR)

The SIU_REACTSR register selects the source of the 16 trigger inputs of the Reaction Module. Each trigger input can come from an eTPU channel output or from an eMIOS channel output.

Address: 0h base + 93Ch offset = 93Ch

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																
R	TS15				TS14				TS13				TS12				TS11				TS10				TS9				TS8				TS7				TS6				TS5				TS4				TS3				TS2				TS1				TS0																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																			
W																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																

SIU_REACTSR field descriptions

Field	Description
0–1 TS15	Trigger Select 15. Selects the source of Reaction Module trigger input 15. 00 eTPU_A Channel 25. 01 eTPU_B Channel 21. 10 eMIOS_0 Channel 7. 11 eMIOS_1 Channel 3.
2–3 TS14	Trigger Select 14. Selects the source of Reaction Module trigger input 14. 00 eTPU_A Channel 24. 01 eTPU_B Channel 20. 10 eMIOS_0 Channel 6. 11 eMIOS_1 Channel 2.
4–5 TS13	Trigger Select 13. Selects the source of Reaction Module trigger input 13. 00 eTPU_A Channel 23. 01 eTPU_B Channel 19. 10 eMIOS_0 Channel 5. 11 eMIOS_1 Channel 1.
6–7 TS12	Trigger Select 12. Selects the source of Reaction Module trigger input 12. 00 eTPU_A Channel 22. 01 eTPU_B Channel 18. 10 eMIOS_0 Channel 4. 11 eMIOS_1 Channel 0.
8–9 TS11	Trigger Select 11. Selects the source of Reaction Module trigger input 11. 00 eTPU_A Channel 21. 01 eTPU_B Channel 17. 10 eTPU_C Channel 31. 11 eMIOS_1 Channel 7.
10–11 TS10	Trigger Select 10. Selects the source of Reaction Module trigger input 10. 00 eTPU_A Channel 20. 01 eTPU_B Channel 16. 10 eTPU_C Channel 30. 11 eMIOS_1 Channel 6.
12–13 TS9	Trigger Select 9. Selects the source of Reaction Module trigger input 9. 00 eTPU_A Channel 19. 01 eTPU_B Channel 15. 10 eTPU_C Channel 29. 11 eMIOS_1 Channel 5.
14–15 TS8	Trigger Select 8. Selects the source of Reaction Module trigger input 8. 00 eTPU_A Channel 18. 01 eTPU_B Channel 14. 10 eTPU_C Channel 28. 11 eMIOS_1 Channel 4.

Table continues on the next page...

SIU_REACTSR field descriptions (continued)

Field	Description
16–17 TS7	Trigger Select 7. Selects the source of Reaction Module trigger input 7. 00 eTPU_A Channel 17. 01 eTPU_B Channel 13. 10 Reserved. 11 eMIOS_0 Channel 3.
18–19 TS6	Trigger Select 6. Selects the source of Reaction Module trigger input 6. 00 eTPU_A Channel 16. 01 eTPU_B Channel 12. 10 Reserved. 11 eMIOS_0 Channel 2.
20–21 TS5	Trigger Select 5. Selects the source of Reaction Module trigger input 5. 00 eTPU_A Channel 15. 01 eTPU_B Channel 11. 10 Reserved. 11 eMIOS_0 Channel 1.
22–23 TS4	Trigger Select 4. Selects the source of Reaction Module trigger input 4. 00 eTPU_A Channel 14. 01 eTPU_B Channel 10. 10 Reserved. 11 eMIOS_0 Channel 0.
24–25 TS3	Trigger Select 3. Selects the source of Reaction Module trigger input 3. 00 eTPU_A Channel 13. 01 eTPU_B Channel 25. 10 eTPU_C Channel 16. 11 eMIOS_0 Channel 19.
26–27 TS2	Trigger Select 2. Selects the source of Reaction Module trigger input 2. 00 eTPU_A Channel 12. 01 eTPU_B Channel 24. 10 eTPU_C Channel 15. 11 eMIOS_0 Channel 18.
28–29 TS1	Trigger Select 1. Selects the source of Reaction Module trigger input 1. 00 eTPU_A Channel 11. 01 eTPU_B Channel 23. 10 eTPU_C Channel 14. 11 eMIOS_0 Channel 17.
30–31 TS0	Trigger Select 0. Selects the source of Reaction Module trigger input 0. 00 eTPU_A Channel 10. 01 eTPU_B Channel 22. 10 eTPU_C Channel 13. 11 eMIOS_0 Channel 16.

8.2.30 Sigma-Delta Gate Selection Register (SIU_SDGATE_SEL)

Each Sigma-Delta ADC supports an input that gates (disable) interrupt and DMA requests. The SIU_SDGATE_SEL register selects the source of the interrupt/DMA gating input. The sources selected by this register come from eTPU output channels. Gating DMA and interrupt requests is useful for applications like knock processing when —outside the knock window—the ADC results are not needed, but the Sigma-Delta ADC is still converting due to its architecture.

Address: 0h base + 940h offset = 940h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	SD_D_GATE_SEL				SD_C_GATE_SEL				SD_B_GATE_SEL				SD_A_GATE_SEL			
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

SIU_SDGATE_SEL field descriptions

Field	Description
0–15 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
16–19 SD_D_GATE_SEL	DMA/Interrupt gating input selection for Sigma-Delta ADC D. 0000 eTPU_A Channel 12. 0001 eTPU_A Channel 30. 0010 eTPU_B Channel 12. 0011 eTPU_B Channel 30. 0100 eTPU_C Channel 10. 0101 eTPU_C Channel 30.
20–23 SD_C_GATE_SEL	DMA/Interrupt gating input selection for Sigma-Delta ADC C. 0000 eTPU_A Channel 11. 0001 eTPU_A Channel 19. 0010 eTPU_B Channel 11. 0011 eTPU_B Channel 19. 0100 eTPU_C Channel 9. 0101 eTPU_C Channel 29.
24–27 SD_B_GATE_SEL	DMA/Interrupt gating input selection for Sigma-Delta ADC B. 0000 eTPU_A Channel 10. 0001 eTPU_A Channel 2. 0010 eTPU_B Channel 10.

Table continues on the next page...

SIU_SDGATE_SEL field descriptions (continued)

Field	Description
	0011 eTPU_B Channel 2. 0100 eTPU_C Channel 8. 0101 eTPU_C Channel 28.
28–31 SD_A_GATE_SEL	DMA/Interrupt gating input selection for Sigma-Delta ADC A. 0000 eTPU_A Channel 9. 0001 eTPU_A Channel 1. 0010 eTPU_B Channel 9. 0011 eTPU_B Channel 1. 0100 eTPU_C Channel 7. 0101 eTPU_C Channel 27.

8.2.31 Sigma-Delta to eTPU_A Connection Configuration Register (SIU_SDETPUA)

This register configures the connection between Sigma-Delta ADCs and eTPU_A input channels. Two ADC signals can be connected to eTPU channel inputs. The first one is a signal that indicates valid data conversion completion. The second one is a watchdog signal that flags a conversion which is outside pre-programmed high/low threshold limits.

Address: 0h base + 944h offset = 944h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0			0					0					0		
W		CH1	CH2		CH4	CH5	CH6	CH7		CH9	CH10	CH11	CH12		CH14	CH15
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R			0		0											0
W	CH16	CH17		CH19											CH30	
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

SIU_SDETPUA field descriptions

Field	Description
0 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

Table continues on the next page...

SIU_SDETPUA field descriptions (continued)

Field	Description
1 CH1	eTPU_A channel 1 input select. Specifies the source of the eTPU_A channel 1 input. 0 eTPU_A channel 1 input pad. 1 Sigma-Delta A conversion limit detection watchdog flag.
2 CH2	eTPU_A channel 2 input select. Specifies the source of the eTPU_A channel 2 input. 0 eTPU_A channel 2 input pad. 1 Sigma-Delta B conversion limit detection watchdog flag.
3 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
4 CH4	eTPU_A channel 4 input select. Specifies the source of the eTPU_A channel 4 input. 0 eTPU_A channel 4 input pad. 1 Sigma-Delta A conversion limit detection watchdog flag.
5 CH5	eTPU_A channel 5 input select. Specifies the source of the eTPU_A channel 5 input. 0 eTPU_A channel 5 input pad. 1 Sigma-Delta B conversion limit detection watchdog flag.
6 CH6	eTPU_A channel 6 input select. Specifies the source of the eTPU_A channel 6 input. 0 eTPU_A channel 6 input pad. 1 Sigma-Delta C conversion limit detection watchdog flag.
7 CH7	eTPU_A channel 7 input select. Specifies the source of the eTPU_A channel 7 input. 0 eTPU_A channel 7 input pad. 1 Sigma-Delta D conversion limit detection watchdog flag.
8 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
9 CH9	eTPU_A channel 9 input select. Specifies the source of the eTPU_A channel 9 input. 0 eTPU_A channel 9 input pad. 1 Sigma-Delta A data valid output signal.
10 CH10	eTPU_A channel 10 input select. Specifies the source of the eTPU_A channel 10 input. 0 eTPU_A channel 10 input pad. 1 Sigma-Delta B data valid output signal.
11 CH11	eTPU_A channel 11 input select. Specifies the source of the eTPU_A channel 11 input. 0 eTPU_A channel 11 input pad. 1 Sigma-Delta C data valid output signal.
12 CH12	eTPU_A channel 12 input select. Specifies the source of the eTPU_A channel 12 input. 0 eTPU_A channel 12 input pad. 1 Sigma-Delta D data valid output signal.
13 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
14 CH14	eTPU_A channel 14 input select. Specifies the source of the eTPU_A channel 14 input.

Table continues on the next page...

SIU_SDETPUA field descriptions (continued)

Field	Description
	0 eTPU_A channel 14 input pad. 1 Sigma-Delta A data valid output signal.
15 CH15	eTPU_A channel 15 input select. Specifies the source of the eTPU_A channel 15 input. 0 eTPU_A channel 15 input pad. 1 Sigma-Delta B data valid output signal.
16 CH16	eTPU_A channel 16 input select. Specifies the source of the eTPU_A channel 16 input. 0 eTPU_A channel 16 input pad. 1 Sigma-Delta C data valid output signal.
17 CH17	eTPU_A channel 17 input select. Specifies the source of the eTPU_A channel 17 input. 0 eTPU_A channel 17 input pad. 1 Sigma-Delta D data valid output signal.
18 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
19 CH19	eTPU_A channel 19 input select. Specifies the source of the eTPU_A channel 19 input. 0 eTPU_A channel 19 input pad. 1 Sigma-Delta C conversion limit detection watchdog flag.
20–29 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
30 CH30	eTPU_A channel 30 input select. Specifies the source of the eTPU_A channel 30 input. 0 Output from multiplex controlled by SIU_IMUX7[MUXSEL14] (eTPUA30 pin or ANB9 pin). 1 Sigma-Delta D conversion limit detection watchdog flag.
31 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

8.2.32 Sigma-Delta to eTPU_B Connection Configuration Register (SIU_SDETPUB)

This register configures the connection between Sigma-Delta ADCs and eTPU_B input channels. Two ADC signals can be connected to eTPU channel inputs. The first one is a signal that indicates valid data conversion completion. The second one is a watchdog signal that flags a conversion which is outside pre-programmed high/low threshold limits.

Address: 0h base + 948h offset = 948h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0			0					0					0		
W		CH1	CH2		CH4	CH5	CH6	CH7		CH9	CH10	CH11	CH12		CH14	CH15
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R			0		0											0
W	CH16	CH17		CH19											CH30	
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

SIU_SDETPUB field descriptions

Field	Description
0 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
1 CH1	eTPU_B channel 1 input select. Specifies the source of the eTPU_B channel 1 input. 0 eTPU_B channel 1 input pad. 1 Sigma-Delta A conversion limit detection watchdog flag.
2 CH2	eTPU_B channel 2 input select. Specifies the source of the eTPU_B channel 2 input. 0 eTPU_B channel 2 input pad. 1 Sigma-Delta B conversion limit detection watchdog flag.
3 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
4 CH4	eTPU_B channel 4 input select. Specifies the source of the eTPU_B channel 4 input. 0 eTPU_B channel 4 input pad. 1 Sigma-Delta A conversion limit detection watchdog flag.
5 CH5	eTPU_B channel 5 input select. Specifies the source of the eTPU_B channel 5 input.

Table continues on the next page...

SIU_SDETPUB field descriptions (continued)

Field	Description
	0 eTPU_B channel 5 input pad. 1 Sigma-Delta B conversion limit detection watchdog flag.
6 CH6	eTPU_B channel 6 input select. Specifies the source of the eTPU_B channel 6 input. 0 eTPU_B channel 6 input pad. 1 Sigma-Delta C conversion limit detection watchdog flag.
7 CH7	eTPU_B channel 7 input select. Specifies the source of the eTPU_B channel 7 input. 0 eTPU_B channel 7 input pad. 1 Sigma-Delta D conversion limit detection watchdog flag.
8 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
9 CH9	eTPU_B channel 9 input select. Specifies the source of the eTPU_B channel 9 input. 0 eTPU_B channel 9 input pad. 1 Sigma-Delta A data valid output signal.
10 CH10	eTPU_B channel 10 input select. Specifies the source of the eTPU_B channel 10 input. 0 eTPU_B channel 10 input pad. 1 Sigma-Delta B data valid output signal.
11 CH11	eTPU_B channel 11 input select. Specifies the source of the eTPU_B channel 11 input. 0 eTPU_B channel 11 input pad. 1 Sigma-Delta C data valid output signal.
12 CH12	eTPU_B channel 12 input select. Specifies the source of the eTPU_B channel 12 input. 0 eTPU_B channel 12 input pad. 1 Sigma-Delta D data valid output signal.
13 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
14 CH14	eTPU_B channel 14 input select. Specifies the source of the eTPU_B channel 14 input. 0 eTPU_B channel 14 input pad. 1 Sigma-Delta A data valid output signal.
15 CH15	eTPU_B channel 15 input select. Specifies the source of the eTPU_B channel 15 input. 0 eTPU_B channel 15 input pad. 1 Sigma-Delta B data valid output signal.
16 CH16	eTPU_B channel 16 input select. Specifies the source of the eTPU_B channel 16 input. 0 eTPU_B channel 16 input pad. 1 Sigma-Delta C data valid output signal.
17 CH17	eTPU_B channel 17 input select. Specifies the source of the eTPU_B channel 17 input. 0 eTPU_B channel 17 input pad. 1 Sigma-Delta D data valid output signal.
18 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

Table continues on the next page...

SIU_SDETPUB field descriptions (continued)

Field	Description
19 CH19	eTPU_B channel 19 input select. Specifies the source of the eTPU_B channel 19 input. 0 eTPU_B channel 19 input pad. 1 Sigma-Delta C conversion limit detection watchdog flag.
20–29 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
30 CH30	eTPU_B channel 30 input select. Specifies the source of the eTPU_B channel 30 input. 0 eTPU_B channel 30 input pad. 1 Sigma-Delta D conversion limit detection watchdog flag.
31 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

8.2.33 Sigma-Delta to eTPU_C Connection Configuration Register (SIU_SDETPUC)

This register configures the connection between Sigma-Delta ADCs and eTPU_C input channels. Two ADC signals can be connected to eTPU channel inputs. The first one is a signal that indicates valid data conversion completion. The second one is a watchdog signal that flags a conversion which is outside pre-programmed high/low threshold limits.

Address: 0h base + 94Ch offset = 94Ch

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0						0				CH10	0				0
W			CH2	CH3	CH4	CH5		CH7	CH8	CH9			CH12	CH13	CH14	
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0										CH25	0				0
W										CH25		CH27	CH28	CH29	CH30	
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

SIU_SDETPUC field descriptions

Field	Description
0–1 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

Table continues on the next page...

SIU_SDETPUC field descriptions (continued)

Field	Description
2 CH2	eTPU_C channel 2 input select. Specifies the source of the eTPU_C channel 2 input. 0 eTPU_C channel 2 input pad. 1 Sigma-Delta A conversion limit detection watchdog flag.
3 CH3	eTPU_C channel 3 input select. Specifies the source of the eTPU_C channel 3 input. 0 Output from multiplex controlled by SIU_IMUX10[MUXSEL3] (eTPUC3 pin or PLLCFG2 pin). 1 Sigma-Delta B conversion limit detection watchdog flag.
4 CH4	eTPU_C channel 4 input select. Specifies the source of the eTPU_C channel 4 input. 0 eTPU_C channel 4 input pad. 1 Sigma-Delta C conversion limit detection watchdog flag.
5 CH5	eTPU_C channel 5 input select. Specifies the source of the eTPU_C channel 5 input. 0 eTPU_C channel 5 input pad. 1 Sigma-Delta D conversion limit detection watchdog flag.
6 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
7 CH7	eTPU_C channel 7 input select. Specifies the source of the eTPU_C channel 7 input. 0 eTPU_C channel 7 input pad. 1 Sigma-Delta A data valid output signal.
8 CH8	eTPU_C channel 8 input select. Specifies the source of the eTPU_C channel 8 input. 0 eTPUC8 pin or ANB23 pin, as selected by SIU_IMUX10[MUXSEL8]. 1 Sigma-Delta B data valid output signal.
9 CH9	eTPU_C channel 9 input select. Specifies the source of the eTPU_C channel 9 input. 0 eTPUC9 pin or ANB22 pin, as selected by SIU_IMUX10[MUXSEL9]. 1 Sigma-Delta C data valid output signal.
10 CH10	eTPU_C channel 10 input select. Specifies the source of the eTPU_C channel 10 input. 0 eTPUC10 pin or ANB21 pin, as selected by SIU_IMUX10[MUXSEL10]. 1 Sigma-Delta D data valid output signal.
11 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
12 CH12	eTPU_C channel 12 input select. Specifies the source of the eTPU_C channel 12 input. 0 eTPUC12 pin or ANB19 pin, as selected by SIU_IMUX10[MUXSEL12]. 1 Sigma-Delta A data valid output signal.
13 CH13	eTPU_C channel 13 input select. Specifies the source of the eTPU_C channel 13 input. 0 eTPUC13 pin or ANB18 pin, as selected by SIU_IMUX10[MUXSEL13]. 1 Sigma-Delta B data valid output signal.
14 CH14	eTPU_C channel 14 input select. Specifies the source of the eTPU_C channel 14 input. 0 eTPUC14 pin or ANB17 pin, as selected by SIU_IMUX10[MUXSEL14]. 1 Sigma-Delta C data valid output signal.

Table continues on the next page...

SIU_SDETPUC field descriptions (continued)

Field	Description
15–24 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
25 CH25	eTPU_C channel 25 input select. Specifies the source of the eTPU_C channel 25 input. 0 eTPU_C channel 25 input pad. 1 Sigma-Delta D data valid output signal.
26 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
27 CH27	eTPU_C channel 27 input select. Specifies the source of the eTPU_C channel 27 input. 0 eTPU_C channel 27 input pad. 1 Sigma-Delta A conversion limit detection watchdog flag.
28 CH28	eTPU_C channel 28 input select. Specifies the source of the eTPU_C channel 28 input. 0 eTPU_C channel 28 input pad. 1 Sigma-Delta B conversion limit detection watchdog flag.
29 CH29	eTPU_C channel 29 input select. Specifies the source of the eTPU_C channel 29 input. 0 eTPU_C channel 29 input pad. 1 Sigma-Delta C conversion limit detection watchdog flag.
30 CH30	eTPU_C channel 30 input select. Specifies the source of the eTPU_C channel 30 input. 0 eTPU_C channel 30 input pad. 1 Sigma-Delta D conversion limit detection watchdog flag.
31 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

8.2.34 Trigger Burst Generator Control Registers for eQADC_A (SIU_TBГ_CR_An)

These registers provide control bits for the Trigger Burst Generators (TBG) connected at the trigger inputs of the 6 CFIFOS of eQADC_A. There is one register for each eQADC_A CFIFO input. The numbering goes from 0 to 5, n=0 corresponding to address 0x950 and CFIFO0, n=5 corresponding to address 0x964 and CFIFO5. See Trigger Burst Generators for detailed information on the TBG feature.

Address: 0h base + 950h offset + (4d × i), where i=0d to 5d

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	GEN	0			CLKSEL		0		TRIGPER							
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	TRIGPER															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

SIU_TBГ_CR_An field descriptions

Field	Description
0 GEN	Global enable bit. This bit enables the operation of the TBG module. If not enabled the TBG is in bypass mode, thus triggers selected by ISEL4-7 are routed directly to the CFIFO input. If enabled, then the TBG behavior depends upon the clock selection CLKSEL bits. 0 TBG is disabled, module operation is transparent regarding the triggers selected by ISEL4-7. 1 TBG is enabled, generating triggers according to ISEL4-7 selection and clock selection.
1–3 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
4–5 CLKSEL	Clock Selection bits. The clock selection bits select three clocks that are used to increment the TBG internal counter: system clock, ADC0 clock, ADC1 clock and no clock. If no clock is select the internal counter remains inactive and no triggers are generated to the corresponding eQADC CFIFO. 00 No clock selected. No triggers are generated independently of ISEL4-7 trigger selection. 01 System clock. TBG increments at system clock rate. 10 ADC0 clock. TBG increments at ADC0 clock rate. 11 ADC1 clock. TBG increments at ADC1 clock rate.
6–7 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
8–31 TRIGPER	Trigger Period. The Trigger period field defines the time interval from one trigger to the next in a trigger burst. The trigger period value is loaded into a TBG internal counter which performs a count down. When zero value is reached one trigger is generated for the corresponding eQADC CFIFO.

8.2.35 Trigger Burst Generator Control Registers for eQADC_B (SIU_TBG_CR_Bn)

These registers provide control bits for the Trigger Burst Generators (TBG) connected at the trigger inputs of the 6 CFIFOS of eQADC_B. There is one register for each eQADC_B CFIFO input. The numbering goes from 0 to 5, n=0 corresponding to address 0x968 and CFIFO0, n=5 corresponding to address 0x97C and CFIFO5. See Trigger Burst Generators for detailed information on the TBG feature.

Address: 0h base + 968h offset + (4d × i), where i=0d to 5d

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	GEN	0			CLKSEL		0		TRIGPER							
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	TRIGPER															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

SIU_TBG_CR_Bn field descriptions

Field	Description
0 GEN	Global enable bit. This bit enables the operation of the TBG module. If not enabled the TBG is in bypass mode, thus triggers selected by ISEL4-7 are routed directly to the CFIFO input. If enabled, then the TBG behavior depends upon the clock selection CLKSEL bits. It is necessary to clear this bit to change the CLKSEL and TRIGPER fields. 0 TBG is disabled, module operation is transparent regarding the triggers selected by ISEL4-7. 1 TBG is enabled, generating triggers according to ISEL4-7 selection and clock selection.
1–3 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
4–5 CLKSEL	Clock Selection bits. The clock selection bits select three clocks that are used to increment the TBG internal counter: peripheral clock (per_clk), ADC0 clock, ADC1 clock and no clock. If no clock is select the internal counter remains inactive and no triggers are generated to the corresponding eQADC CFIFO. NOTE: It is necessary to clear the global enable bit GEN when changing this field. 00 No clock selected. No triggers are generated independently of ISEL4-7 trigger selection. 01 Peripheral clock (per_clk). TBG increments at per_clk rate. 10 ADC0 clock. TBG increments at ADC0 clock rate. 11 ADC1 clock. TBG increments at ADC1 clock rate.
6–7 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
8–31 TRIGPER	Trigger Period. The Trigger period field defines the time interval from one trigger to the next in a trigger burst. The trigger period value is loaded into a TBG internal counter which performs a count down. When zero value is reached one trigger is generated for the corresponding eQADC CFIFO. NOTE: It is recommended to clear the global enable bit GEN when changing this field.

8.2.36 External Clock Control Register (SIU_ECCR)

The SIU_ECCR controls the timing relationship between the system clock and the external clocks ENGCLK and CLKOUT.

Address: 0h base + 984h offset = 984h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	ENGDIV								ECSS	0				0	EBDF	
W																
Reset	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	1

SIU_ECCR field descriptions

Field	Description
0–15 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
16–23 ENGDIV	Engineering Clock Division Factor. The ENGDIV field specifies the frequency ratio between the platform clock (plat_clk) and ENGCLK. The ENGCLK frequency is divided from the platform clock frequency according to the following equation: $EngineeringClockFrequency = \frac{PlatformClockFrequency}{(ENGDIV \times 2)}$ Setting ENGDIV to 0 makes the ENGCLK frequency equal to the platform clock frequency. NOTE: Maximum ENGCLK frequency is limited by the pad performance. Refer to the Electrical Specifications in the Data Sheet document for this device.
24 ECSS	Engineering clock source select. The ECSS bit selects the source clock to be divided to generate the ENGCLK clock. 0 The system clock is the source of the ENGCLK. 1 The crystal oscillator clock is the source of ENGCLK.
25–27 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
28 EBTS	External Bus Tap Select. Changes the phase relationship between the system clock and CLKOUT. Changing the phase relationship so that CLKOUT is advanced in relation to system clock increases the output hold time of the external bus signals to a non-zero value. It also increases the output delay times, increases the input hold times to non-zero values, and decreases the input setup times. Refer to the Data Sheet document for how the EBTS bit affects the electrical specifications.

Table continues on the next page...

SIU_ECCR field descriptions (continued)

Field	Description
	NOTE: The EBTS bit must not be modified while an external bus transaction is in progress. 0 External bus signals have zero output hold times. 1 External bus signals have non-zero output hold times.
29 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
30–31 EBDF	External Bus Division Factor. Specifies the frequency ratio between the system clock and CLKOUT. The EBDF field must not be changed during an external bus access or while an access is pending. 00 Divide by 1. 01 Divide by 2. 10 Divide by 3. 11 Divide by 4.

8.2.37 System Clock Register (SIU_SYSDIV)

The SIU_SYSDIV register provides clock multiplexing control and clock frequency division factors to configure the system clocks. Please refer to the clocking section for a diagram showing how the fields of this register are used to control the various clock muxes and dividers.

NOTE

Do not change the contents of this register while the FEC is operating, otherwise transactions in the crossbar may be corrupted.

Address: 0h base + 9A0h offset = 9A0h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	LCK	0		PLLOSEL	0		PLL1SEL	0		0				0		
W											FMPERDIV					PERDIV
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	MCANSEL	0				0				0					0	
W				SYSCLKSEL				ETPUDIV					SYSCLKDIV			PCSEN
Reset	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0

SIU_SYSDIV field descriptions

Field	Description
0 LCK	Write Lock Bit. When the LCK is written with a value of 1, writes to all other bits and fields in the SIU_SYSDIV register are disabled. The LCK bit can be set/cleared by software at any time. 0 SIU_SYSDIV register is unlocked and writes are permitted. 1 SIU_SYSDIV register is locked, and only writes to the LCK bit are permitted.
1–2 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
3 PLL0SEL	PLL0 Select. Selects the clock source for PLL0. 0 The crystal oscillator (XOSC) is the clock source for PLL0. 1 The internal RC oscillator (IRC) is the clock source for PLL0.
4 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
5 PLL1SEL	PLL1 Select. Selects the clock source for PLL1. 0 The crystal oscillator (XOSC) is the clock source for PLL1. 1 The PHI1 output of PLL0 is the clock source for PLL1.
6 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
7 PERCLKSEL	Peripheral Clock Select. Selects the clock source for eTPU and other communication peripherals that require non-FM clock for their protocol engines. 0 Clock is connected to the FM clock domain (output of the SYSCLKDIV divider), for legacy support. 1 Clock is connected to the non-FM clock domain (output of PLL0).
8–9 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
10–11 FMPERDIV	FM Clock Peripheral Divider. Selects the divider value for the clock that feeds the memory mapped portion of the peripheral interfaces, and also feeds peripheral engines that can tolerate FM modulation. CAUTION: For proper device operation, always use the FMPERDIV setting 00b for divide by 2. 00 Divide by 2. 01 Divide by 4. 10 Divide by 8. 11 Reserved.
12–13 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
14–15 PERDIV	Non-FM Clock Peripheral Divider. Selects the divider value for the clock that feeds protocol engines of peripherals that must be clocked by a stable non-FM clock. 00 Divide by 2. 01 Divide by 4. 10 Divide by 8. 11 Reserved.
16 MCANSEL	MCAN Select. Selects the clock source for both MCAN modules. 0 The crystal oscillator (XOSC) is the clock source for the MCAN modules. 1 The non-FM peripheral clock is the clock source for the MCAN modules.

Table continues on the next page...

SIU_SYSDIV field descriptions (continued)

Field	Description
17 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
18–19 SYSCLKSEL	System Clock Select. The SYSCLKSEL field directly selects the source of the system clock if the Progressive Clock Switch (PCS) is disabled. If the PCS is enabled, then the SYSCLKSEL field is a request to change the clock, and the actual changing is controlled by the PCS. In this case, information about the actual clock source driving the system clock is available in the SIU_PCSIFR[PCSMS] field. Note that if the PCS fails to switch (because, for example, it was programmed incorrectly), then the system clock will not switch to the clock requested by SYSCLKSEL, and SIU_PCSIFR[PCSMS] will still indicate the previous clock. 00 The system clock is driven by the 16 MHz Internal RC Oscillator (IRC). 01 The system clock is driven by the external oscillator (XOSC). 10 The system clock is driven by the FM PLL (PLL1). 11 The system clock is driven by the non-FM PLL (PLL0).
20–22 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
23 ETPUDIV	eTPU Clock Divider. Selects the divider value for the eTPU clock. 0 Divide by 2. 1 Divide by 1.
24–26 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
27–29 SYSCLKDIV	System Clock Divider. Selects the divider value for the system clock coming out of PLL1. Note that the SYSCLKDIV divider is required in addition to the PLL output divider to allow other sources for the system clock (IRC, OSC) to be divided down to slowest frequencies to enable very low power modes. 000 Divide by 2. 001 Divide by 4. 010 Divide by 8. 011 Divide by 16. 100 Divide by 1. 101 Divide by 1. 110 Divide by 1. 111 Divide by 1.
30 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
31 PCSEN	Progressive Clock Switch Enable. Enables or disables the progressive clock switching (PCS) feature on the system clock. 0 PCS disabled. 1 PCS enabled.

8.2.38 Halt Register 1 (SIU_HLT1)

This register is used to dispatch a request to various modules to halt their clock inputs. Each register bit drives a separate halt request signal. Modules respond differently to the halt request. The CPUs will allow their clock inputs to be halted only after the execution of the WAIT instruction. Most other modules will wait for internal activities to finish before allowing their clock inputs to be halted, but a few modules will allow the clocks to be halted immediately.

Some peripherals have an MDIS (module disable) bit in the module control register that can be set to disable the module clock, reducing power consumption. In most cases the peripheral registers are still readable and writeable. However, using the SIU_HALT1 register also disables the read/write functions of the disabled peripheral registers for additional power saving.

When the clocks of all CPUs have been halted and all cross-bar masters are idle, then the clocks of the following modules are also halted automatically: cross-bar, peripheral bridges, system RAM, DMA, STM and Semaphores. The interrupt controller and SWT_A/SWT_B clocks are never stopped.

Address: 0h base + 9A4h offset = 9A4h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	CORE0	CORE1	0	CSE	ETPUC	ETPUA	NPC	EBI	EQADCB	EQADCA	EMIOS0	DECFIL	EMIOS1	PIT	0	
W																
Reset	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	FLEXCAND	FLEXCANC	FLEXCANB	FLEXCANA	DSPID	DSPIC	DSPIB	DSPIA	DSPIE	0	ESCIF	ESCIE	ESCID	ESCIC	ESCIB	ESCIA
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

SIU_HLT1 field descriptions

Field	Description
0 CORE0	Halt Core 0. Halts the clock inputs of Core 0 after the execution of the WAIT instruction.
0	No halt request.
1	Dispatch halt request.

Table continues on the next page...

SIU_HLT1 field descriptions (continued)

Field	Description
1 CORE1	Halt Core 1. Halt the clock inputs of Core 1 after the execution of the WAIT instruction. 0 No halt request. 1 Dispatch halt request.
2 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
3 CSE	Halt CSE. Halt the clock inputs of the Cryptographic Services Engine. The clocks are halted immediately after this bit is set, therefore software must guarantee that the CSE module can be stopped prior to asserting the halt request. 0 No halt request. 1 Dispatch halt request.
4 ETPUC	Halt eTPU C. Dispatch a halt request to Enhanced Timing Processing Unit C and to the Reaction Module. These modules will hold the request until all internal activities are concluded, then they will allow their clock inputs to be halted. 0 No halt request. 1 Dispatch halt request.
5 ETPUA	Halt eTPU A/B. Dispatch a halt request to Enhanced Timing Processing Unit A/B. The module will hold the request until all internal activities are concluded, then it will allow its clock inputs to be halted. 0 No halt request. 1 Dispatch halt request.
6 NPC	Halt NPC. Dispatch a halt request to the Nexus Port Controller. The module will hold the request until all internal activities are concluded, then it will allow its clock inputs to be halted. NOTE: When this bit is 1 and the NPC clock is halted, if any of the following Nexus clients gets out of the IDLE state, the NPC clock is re-enabled temporarily: • Core0 • Core1 • eDMA_A • eDMA_B • SIPI The NPC clock is re-enabled only during the period when any of these clients is out of the IDLE state. When all of these clients resume the IDLE state, the NPC clock is halted again automatically. 0 No halt request. 1 Dispatch halt request.
7 EBI	Halt EBI. Dispatch a halt request to the External Bus Interface module. The module will hold the request until all internal activities are concluded, then it will allow its clock inputs to be halted. 0 No halt request. 1 Dispatch halt request.
8 EQADCB	Halt eQADC B. Dispatch a halt request to the Enhanced Queued Analog-to-Digital Converter B. The module will hold the request until all internal activities are concluded, then it will allow its clock inputs to be halted. This halt request prompts eQADC_B to enter the module's Stop mode. 0 No halt request. 1 Dispatch halt request.

Table continues on the next page...

SIU_HLT1 field descriptions (continued)

Field	Description
9 EQADCA	Halt eQADC A. Dispatch a halt request to the Enhanced Queued Analog-to-Digital Converter A. The module will hold the request until all internal activities are concluded, then it will allow its clock inputs to be halted. This halt request prompts eQADC_A to enter the module's Stop mode. 0 No halt request. 1 Dispatch halt request.
10 EMIOS0	Halt eMIOS 0. Halt the clock inputs of the Enhanced Modular I/O System 0. The clocks are halted immediately after this bit is set, therefore software must guarantee that the eMIOS_0 module can be stopped prior to asserting the halt request. 0 No halt request. 1 Dispatch halt request.
11 DECFIL	Halt DECFIL. Dispatch a halt request to all Decimation Filter modules. The modules will hold the request until all internal activities are concluded, then they will allow their clock inputs to be halted. 0 No halt request. 1 Dispatch halt request.
12 EMIOS1	Halt eMIOS 1. Halt the clock inputs of the Enhanced Modular I/O System 1. The clocks are halted immediately after this bit is set, therefore software must guarantee that the eMIOS_1 module can be stopped prior to asserting the halt request. 0 No halt request. 1 Dispatch halt request.
13 PIT	Halt PIT. Dispatch a halt request to the Periodic Interval Timer. The module will hold the request until all internal activities are concluded, then it will allow its clock inputs to be halted. 0 No halt request. 1 Dispatch halt request.
14–15 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
16 FLEXCAND	Halt FlexCAN D. Dispatch a halt request to the Flexible Controller Area Network D. The module will hold the request until all internal activities are concluded, then it will allow its clock inputs to be halted. 0 No halt request. 1 Dispatch halt request.
17 FLEXCANC	Halt FlexCAN C. Dispatch a halt request to the Flexible Controller Area Network C. The module will hold the request until all internal activities are concluded, then it will allow its clock inputs to be halted. 0 No halt request. 1 Dispatch halt request.
18 FLEXCANB	Halt FlexCAN B. Dispatch a halt request to the Flexible Controller Area Network B. The module will hold the request until all internal activities are concluded, then it will allow its clock inputs to be halted. 0 No halt request. 1 Dispatch halt request.
19 FLEXCANA	Halt FlexCAN A. Dispatch a halt request to the Flexible Controller Area Network A. The module will hold the request until all internal activities are concluded, then it will allow its clock inputs to be halted. 0 No halt request. 1 Dispatch halt request.

Table continues on the next page...

SIU_HLT1 field descriptions (continued)

Field	Description
20 DSPID	Halt DSPI D. Dispatch a halt request to the Deserial/Serial Peripheral Interface D. The module will hold the request until all internal activities are concluded, then it will allow its clock inputs to be halted. 0 No halt request. 1 Dispatch halt request.
21 DSPIC	Halt DSPI C. Dispatch a halt request to the Deserial/Serial Peripheral Interface C. The module will hold the request until all internal activities are concluded, then it will allow its clock inputs to be halted. 0 No halt request. 1 Dispatch halt request.
22 DSPIB	Halt DSPI B. Dispatch a halt request to the Deserial/Serial Peripheral Interface B. The module will hold the request until all internal activities are concluded, then it will allow its clock inputs to be halted. 0 No halt request. 1 Dispatch halt request.
23 DSPIA	Halt DSPI A. Dispatch a halt request to the Deserial/Serial Peripheral Interface A. The module will hold the request until all internal activities are concluded, then it will allow its clock inputs to be halted. 0 No halt request. 1 Dispatch halt request.
24 DSPIE	Halt DSPI E. Dispatch a halt request to the Deserial/Serial Peripheral Interface E. The module will hold the request until all internal activities are concluded, then it will allow its clock inputs to be halted. 0 No halt request. 1 Dispatch halt request.
25 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
26 ESCIF	Halt eSCI F. Dispatch a halt request to the Enhanced Serial Communication Interface F. The module will hold the request until all internal activities are concluded, then it will allow its clock inputs to be halted. 0 No halt request. 1 Dispatch halt request.
27 ESCIE	Halt eSCI E. Dispatch a halt request to the Enhanced Serial Communication Interface E. The module will hold the request until all internal activities are concluded, then it will allow its clock inputs to be halted. 0 No halt request. 1 Dispatch halt request.
28 ESCID	Halt eSCI D. Dispatch a halt request to the Enhanced Serial Communication Interface D. The module will hold the request until all internal activities are concluded, then it will allow its clock inputs to be halted. 0 No halt request. 1 Dispatch halt request.
29 ESCIC	Halt eSCI C. Dispatch a halt request to the Enhanced Serial Communication Interface C. The module will hold the request until all internal activities are concluded, then it will allow its clock inputs to be halted. 0 No halt request. 1 Dispatch halt request.
30 ESCIB	Halt eSCI B. Dispatch a halt request to the Enhanced Serial Communication Interface B. The module will hold the request until all internal activities are concluded, then it will allow its clock inputs to be halted.

Table continues on the next page...

SIU_HLT1 field descriptions (continued)

Field	Description
	0 No halt request. 1 Dispatch halt request.
31 ESCIA	Halt eSCI A. Dispatch a halt request to the Enhanced Serial Communication Interface A. The module will hold the request until all internal activities are concluded, then it will allow its clock inputs to be halted. 0 No halt request. 1 Dispatch halt request.

8.2.39 Halt Acknowledge Register 1 (SIU_HLTACK1)

The SIU_HLTACK1 bits indicate that the request to halt the modules via the respective SIU_HLT1 bits have completed and the clock inputs to the modules have been disabled.

Address: 0h base + 9A8h offset = 9A8h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	CORE0	CORE1	0	CSE	ETPUC	ETPUA	NPC	EBI	EQADCB	EQADCA	EMIOS0	DECFIL	EMIOS1	PIT	0	
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	FLEXCAND	FLEXCANC	FLEXCANB	FLEXCANA	DSPID	DSPIC	DSPIB	DSPIA	DSPIE	0	ESCIF	ESCIE	ESCID	ESCIC	ESCIB	ESCIA
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

SIU_HLTACK1 field descriptions

Field	Description
0 CORE0	Halt Acknowledge Core 0.

Table continues on the next page...

SIU_HLTACK1 field descriptions (continued)

Field	Description
	0 Not halted. 1 Halted.
1 CORE1	Halt Acknowledge Core 1. 0 Not halted. 1 Halted.
2 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
3 CSE	Halt Acknowledge CSE. 0 Not halted. 1 Halted.
4 ETPUC	Halt Acknowledge eTPU C and Reaction Module. 0 Not halted. 1 Both eTPU C and Reaction Modules are Halted.
5 ETPUA	Halt Acknowledge eTPU A/B. 0 Not halted. 1 Halted.
6 NPC	Halt Acknowledge NPC. 0 Not halted. 1 Halted.
7 EBI	Halt Acknowledge EBI. 0 Not halted. 1 Halted.
8 EQADCB	Halt Acknowledge eQADC B. 0 Not halted. 1 Halted.
9 EQADCA	Halt Acknowledge eQADC A. 0 Not halted. 1 Halted.
10 EMIOS0	Halt Acknowledge eMIOS 0. 0 Not halted. 1 Halted.
11 DECFIL	Halt Acknowledge DECFIL. 0 Not halted. 1 Halted.
12 EMIOS1	Halt Acknowledge eMIOS 1. 0 Not halted. 1 Halted.

Table continues on the next page...

SIU_HLTACK1 field descriptions (continued)

Field	Description
13 PIT	Halt Acknowledge PIT. 0 Not halted. 1 Halted.
14–15 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
16 FLEXCAND	Halt Acknowledge FlexCAN D. 0 Not halted. 1 Halted.
17 FLEXCANC	Halt Acknowledge FlexCAN C. 0 Not halted. 1 Halted.
18 FLEXCANB	Halt Acknowledge FlexCAN B. 0 Not halted. 1 Halted.
19 FLEXCANA	Halt Acknowledge FlexCAN A. 0 Not halted. 1 Halted.
20 DSPID	Halt Acknowledge DSPI D. 0 Not halted. 1 Halted.
21 DSPIC	Halt Acknowledge DSPI C. 0 Not halted. 1 Halted.
22 DSPIB	Halt Acknowledge DSPI B. 0 Not halted. 1 Halted.
23 DSPIA	Halt Acknowledge DSPI A. 0 Not halted. 1 Halted.
24 DSPIE	Halt Acknowledge DSPI E. 0 Not halted. 1 Halted.
25 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
26 ESCIF	Halt Acknowledge eSCI F. 0 Not halted. 1 Halted.

Table continues on the next page...

SIU_HLTACK1 field descriptions (continued)

Field	Description
27 ESCIE	Halt Acknowledge eSCI E. 0 Not halted. 1 Halted.
28 ESCID	Halt Acknowledge eSCI D. 0 Not halted. 1 Halted.
29 ESCIC	Halt Acknowledge eSCI C. 0 Not halted. 1 Halted.
30 ESCIB	Halt Acknowledge eSCI B. 0 Not halted. 1 Halted.
31 ESCIA	Halt Acknowledge eSCI A. 0 Not halted. 1 Halted.

8.2.40 Core0 Reset Vector Register (SIU_RSTVEC0)

This register contains the reset vector, reset control and instruction execution type selection for Core0. The RST bit is qualified by the CORE0 halt bit in the SIU_HLT1 register, such that both must be 1 for a core reset to occur. The order in which the RST and core halt bits are asserted does not matter. After both halt and RST bits are asserted, internal logic places a request to halt the corresponding core clocks, but the core may continue to run until such time that it terminates any pending crossbar master bus cycles and interrupt controller interaction that might prevent their use by other bus masters. When all pending transactions finish, the core will be in reset state with clock stopped. Exit of reset state is done by negating either SIU_HLT1[CORE0] or the RST bit. Negating just one is enough to remove the core from reset state.

The reset value of the halt and RST bits for Core0 is 0 by default to allow Core0 to execute out of reset. The halt and RST bits for Core1 are 1 by default to prevent Core1 from executing out of reset.

Any attempt to reset both cores by setting both RST bits will result in an immediate system reset, with the cause of the reset recorded in the SIU_RSR[CPURS] bit.

NOTE

The halt acknowledge bit in the SIU_HLTACK1 register will not be asserted for a core in reset state. Even though setting the

halt bit is part of the procedure to put a core in reset state, the halt acknowledge bit is only asserted when the WAIT instruction is executed and the core is stopped, but not in reset state.

Address: 0h base + 9ACh offset = 9ACh

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	RSTVEC															
W																
Reset	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	RSTVEC														RST	VLE
W																
Reset	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0	0

SIU_RSTVEC0 field descriptions

Field	Description
0–29 RSTVEC	Reset Vector. The RSTVEC value determines the initial program counter upon exiting reset. The MMU TLB entry 0 effective and real page number (EPN/RPN) is automatically loaded with RSTVEC[0:21] when reset is released and the TLB entry size is set to 4KB. In the e200z759 Core Reference Manual the reset vector is referred to as p_rstbase[0-21].
30 RST	Controls the assertion of RESET to the core. Writing 1 to this bit causes the core to enter reset, provided the respective SIU_HLT1 bit is also set. Reads of this bit indicate whether the core is being held in reset. 0 Core is not in reset. 1 Reset request.
31 VLE	VLE Select. The VLE bit selects whether the core executes VLE or Book E code at the reset vector address. 0 Book E. 1 VLE.

8.2.41 Core1 Reset Vector Register (SIU_RSTVEC1)

This register contains the reset vector, reset control and instruction execution type selection for Core1. The RST bit is qualified by the CORE1 halt bit in the SIU_HLT1 register, such that both must be 1 for a core reset to occur. The order in which the RST and core halt bits are asserted does not matter. After both halt and RST bits are asserted, internal logic places a request to halt the corresponding core clocks, but the core may continue to run until such time that it terminates any pending crossbar master bus cycles and interrupt controller interaction that might prevent their use by other bus masters. When all pending transactions finish, the core will be in reset state with clock stopped. Exit of reset state is done by negating either SIU_HLT1[CORE1] or the RST bit. Negating just one is enough to remove the core from reset state.

The reset value of the halt and RST bits for Core0 is 0 by default to allow Core0 to execute out of reset. The halt and RST bits for Core1 are 1 by default to prevent Core1 from executing out of reset.

Any attempt to reset both cores by setting both RST bits will result in an immediate system reset, with the cause of the reset recorded in the SIU_RSR[CPURS] bit.

NOTE

The halt acknowledge bit in the SIU_HLTACK1 register will not be asserted for a core in reset state. Even though setting the halt bit is part of the procedure to put a core in reset state, the halt acknowledge bit is only asserted when the WAIT instruction is executed and the core is stopped, but not in reset state.

Address: 0h base + 9B0h offset = 9B0h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	RSTVEC															
W																
Reset	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	RSTVEC														RST	VLE
W																
Reset	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0

SIU_RSTVEC1 field descriptions

Field	Description
0–29 RSTVEC	Reset Vector. The RSTVEC value determines the initial program counter upon exiting reset. The MMU TLB entry 0 effective and real page number (EPN/RPN) is automatically loaded with RSTVEC[0:21] when reset is released and the TLB entry size is set to 4KB. In the e200z759 Core Reference Manual the reset vector is referred to as p_rstbase[0-21].
30 RST	Controls the assertion of RESET to the core. Writing 1 to this bit causes the core to enter reset, provided the respective SIU_HLT1 bit is also set. Reads of this bit indicate whether the core is being held in reset. 0 Core is not in reset. 1 Reset request.
31 VLE	VLE Select. The VLE bit selects whether the core executes VLE or Book E code at the reset vector address. 0 Book E. 1 VLE.

8.2.42 Core0 PID mapping control register (SIU_C0PID)

This register allows an external tool to modify the logical to physical address mapping without interrupting normal application code execution in either core. An optional synchronization mechanism using the nex_wevto[2] watchpoint event output from each core allows the mapping that is selected by the tool to change deterministically, when a specified instruction address is reached or a specified load/store address is accessed. If the synchronization mechanism is not enabled, a write to the register is propagated immediately to the MMU.

Address: 0h base + 9B4h offset = 9B4h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15															
R	EXT_PID_EN	EXT_PID_SYNC	0																												
W																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0															
Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31															
R	0																													EXT_PID	
W																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0															

SIU_C0PID field descriptions

Field	Description
0 EXT_PID_EN	External PID Selection Enable. Enable the values of the EXT_PID field to be routed to the MMU. 0 The contents of this register are not used to select the alternate MMU mapping. 1 The contents of the EXT_PID field are used to select the alternate MMU mapping.
1 EXT_PID_SYNC	External PID Synchronization. This bit configures whether the Nexus Watchpoint Event 2 is used or not to transfer the EXT_PID values to the MMU. 0 The Nexus Watchpoint Event 2 does not transfer the EXT_PID field to the MMU. 1 The Nexus Watchpoint Event 2 transfers the EXT_PID field to the MMU.
2–29 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
30–31 EXT_PID	External PID. These are the external PID bits that are routed by the MMU to implement an alternate logical to physical address mapping under control of an external tool.

8.2.43 Core1 PID mapping control register (SIU_C1PID)

This register allows an external tool to modify the logical to physical address mapping without interrupting normal application code execution in either core. An optional synchronization mechanism using the nex_wevto[2] watchpoint event output from each core allows the mapping that is selected by the tool to change deterministically, when a specified instruction address is reached or a specified load/store address is accessed. If the synchronization mechanism is not enabled, a write to the register is propagated immediately to the MMU.

Address: 0h base + 9B8h offset = 9B8h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	EXT_PID_EN	EXT_PID_SYNC	0													
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31															
R	0																													EXT_PID	
W																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0															

SIU_C1PID field descriptions

Field	Description
0 EXT_PID_EN	External PID Selection Enable. Enable the values of the EXT_PID field to be routed to the MMU. 0 The contents of this register are not used to select the alternate MMU mapping. 1 The contents of the EXT_PID field are used to select the alternate MMU mapping.
1 EXT_PID_SYNC	External PID Synchronization. This bit configures whether the Nexus Watchpoint Event 2 is used or not to transfer the EXT_PID values to the MMU. 0 The Nexus Watchpoint Event 2 does not transfer the EXT_PID field to the MMU. 1 The Nexus Watchpoint Event 2 transfers the EXT_PID field to the MMU.
2–29 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
30–31 EXT_PID	External PID. These are the external PID bits that are routed by the MMU to implement an alternate logical to physical address mapping under control of an external tool.

8.2.44 Halt Register 2 (SIU_HLT2)

This register is used to dispatch a request to various modules to halt their clock inputs. Each register bit drives a separate halt request signal. Modules respond differently to the halt request. Most modules will wait for internal activities to finish before allowing their clock inputs to be halted, but a few modules will allow the clocks to be halted immediately.

Some peripherals have an MDIS (module disable) bit in the module control register that can be set to disable the module clock, reducing power consumption. In most cases the peripheral registers are still readable and writeable. However, using the SIU_HALT2 register also disables the read/write functions of the disabled peripheral registers for additional power saving.

When the clocks of all CPUs have been halted and all cross-bar masters are idle, then the clocks of the following modules are also halted automatically: cross-bar, peripheral bridges, system RAM, DMA, STM and Semaphores. The interrupt controller and SWT_A/SWT_B clocks are never stopped.

Address: 0h base + 9BCh offset = 9BCh

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0												SDD	SDC	SDB	SDA
W	FEC															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	SIPI	CRC	STCU	0			SRX1	SRX0	0		PSI5B	PSI5A	0		MCANB	MCANA
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

SIU_HLT2 field descriptions

Field	Description
0 FEC	Halt FEC. Halt the clock inputs of the Fast Ethernet Controller. The clocks are halted immediately after this bit is set, therefore software must guarantee that the FEC module can be stopped prior to asserting the halt request. 0 No halt request. 1 Dispatch halt request.
1–11 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

Table continues on the next page...

SIU_HLT2 field descriptions (continued)

Field	Description
12 SDD	Halt Sigma Delta D. Dispatch a halt request to Sigma Delta D. The module will hold the request until all internal activities are concluded, then it will allow its clock inputs to be halted (digital and analog). 0 No halt request. 1 Dispatch halt request.
13 SDC	Halt Sigma Delta C. Dispatch a halt request to Sigma Delta C. The module will hold the request until all internal activities are concluded, then it will allow its clock inputs to be halted (digital and analog). 0 No halt request. 1 Dispatch halt request.
14 SDB	Halt Sigma Delta B. Dispatch a halt request to Sigma Delta B. The module will hold the request until all internal activities are concluded, then it will allow its clock inputs to be halted (digital and analog). 0 No halt request. 1 Dispatch halt request.
15 SDA	Halt Sigma Delta A. Dispatch a halt request to Sigma Delta A. The module will hold the request until all internal activities are concluded, then it will allow its clock inputs to be halted (digital and analog). 0 No halt request. 1 Dispatch halt request.
16 SIPI	Halt SIPI / LFAST. Halt the clocks to the SIPI and LFAST modules. The clocks are halted immediately after this bit is set, therefore software must guarantee that the SIPI and LFAST modules can be stopped prior to asserting the halt request. 0 No halt request. 1 Dispatch halt request.
17 CRC	Halt CRC. Halt the clocks to the CRC. The clocks are halted immediately after this bit is set, therefore software must guarantee that the CRC can be stopped prior to asserting the halt request. 0 No halt request. 1 Dispatch halt request.
18 STCU	Halt STCU. Halt the clocks to the STCU. The clocks are halted immediately after this bit is set, therefore software must guarantee that the STCU can be stopped prior to asserting the halt request. 0 No halt request. 1 Dispatch halt request.
19–21 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
22 SRX1	Halt SENT 1. Dispatch a halt request to the SENT 1. The module will hold the request until all internal activities are concluded, then it will allow its clock inputs to be halted. 0 No halt request. 1 Dispatch halt request.
23 SRX0	Halt SENT 0. Dispatch a halt request to the SENT 0. The module will hold the request until all internal activities are concluded, then it will allow its clock inputs to be halted. 0 No halt request. 1 Dispatch halt request.
24–25 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

Table continues on the next page...

SIU_HLT2 field descriptions (continued)

Field	Description
26 PSI5B	Halt PSI5 B. Dispatch a halt request to PSI5 B. The module will hold the request until all internal activities are concluded, then they will allow its clock inputs to be halted. 0 No halt request. 1 Dispatch halt request.
27 PSI5A	Halt PSI5 A. Dispatch a halt request to PSI5 A. The module will hold the request until all internal activities are concluded, then they will allow its clock inputs to be halted. 0 No halt request. 1 Dispatch halt request.
28–29 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
30 MCANB	Halt MCAN B. Dispatch a halt request to MCAN B. The module will hold the request until all internal activities are concluded, then it will allow its clock inputs to be halted. 0 No halt request. 1 Dispatch halt request.
31 MCANA	Halt MCAN A. Dispatch a halt request to MCAN A. The module will hold the request until all internal activities are concluded, then it will allow its clock inputs to be halted. 0 No halt request. 1 Dispatch halt request.

8.2.45 Halt Acknowledge Register 2 (SIU_HLTACK2)

The SIU_HLTACK2 bits indicate that the request to halt the modules via the respective SIU_HLT2 bits have completed and the clock inputs to the modules have been disabled.

Address: 0h base + 9C0h offset = 9C0h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	FEC	0											SDD	SDC	SDB	SDA
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	SIPI	CRC	STCU	0			SRX1	SRX0	0		PSI5B	PSI5A	0		MCANB	MCANA
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

SIU_HLTACK2 field descriptions

Field	Description
0 FEC	Halt Acknowledge FEC. 0 Not halted. 1 Halted.

Table continues on the next page...

SIU_HLTACK2 field descriptions (continued)

Field	Description
1–11 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
12 SDD	Halt Acknowledge Sigma Delta D. 0 Not halted. 1 Halted.
13 SDC	Halt Acknowledge Sigma Delta C. 0 Not halted. 1 Halted.
14 SDB	Halt Acknowledge Sigma Delta B. 0 Not halted. 1 Halted.
15 SDA	Halt Acknowledge Sigma Delta A. 0 Not halted. 1 Halted.
16 SIPI	Halt Acknowledge SIPI / LFAST. 0 Not halted. 1 Halted.
17 CRC	Halt Acknowledge CRC. 0 Not halted. 1 Halted.
18 STCU	Halt Acknowledge STCU. 0 Not halted. 1 Halted.
19–21 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
22 SRX1	Halt Acknowledge SENT 1. 0 Not halted. 1 Halted.
23 SRX0	Halt Acknowledge SENT 0. 0 Not halted. 1 Halted.
24–25 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
26 PSI5B	Halt Acknowledge PSI5 B. 0 Not halted. 1 Halted.
27 PSI5A	Halt Acknowledge PSI5 A.

Table continues on the next page...

SIU_HLTACK2 field descriptions (continued)

Field	Description
	0 Not halted. 1 Halted.
28–29 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
30 MCANB	Halt Acknowledge MCAN B. 0 Not halted. 1 Halted.
31 MCANA	Halt Acknowledge MCAN A. 0 Not halted. 1 Halted.

8.2.46 Sigma-Delta Clock Configuration Register (SIU_SDCLKCFG)

The SIU_SDCLKCFG register defines the clock division factor to generate the clock to the Sigma-Delta ADCs. [Figure 19-1](#) shows the SDDIV field in the clock generation scheme.

Address: 0h base + 9C4h offset = 9C4h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	LCK	0														
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0								SDDIV							
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

SIU_SDCLKCFG field descriptions

Field	Description
0 LCK	Write Lock Bit. When the LCK is written with a value of 1, writes to all other bits and fields in the SIU_SDCLKCFG register are disabled. The LCK bit can be set/cleared by software at any time. 0 SIU_SDCLKCFG register is unlocked and writes are permitted. 1 SIU_SDCLKCFG register is locked, and only writes to the LCK bit are permitted.
1–24 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
25–31 SDDIV	Sigma Delta Clock Divider. Selects the divider value for the clock that feeds the Sigma Delta ADCs, according to the following formula: DivisionFactor = SDDIV + 1

8.2.47 LFAST Clock Configuration Register (SIU_LFCLKCFG)

The SIU_LFCLKCFG register defines the clock division factor to generate the clock to the LFAST module. [Figure 19-1](#) shows the LFDIV and LFCLKSEL fields in the clock generation scheme.

Address: 0h base + 9C8h offset = 9C8h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	LCK								0							
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0								0							
W			LFCLKSEL												LFDIV	
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

SIU_LFCLKCFG field descriptions

Field	Description
0 LCK	Write Lock Bit. When the LCK is written with a value of 1, writes to all other bits and fields in the SIU_LFCLKCFG register are disabled. The LCK bit can be set/cleared by software at any time. 0 SIU_LFCLKCFG register is unlocked and writes are permitted. 1 SIU_LFCLKCFG register is locked, and only writes to the LCK bit are permitted.
1–17 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
18–19 LFCLKSEL	LFAST Clock Select. This field selects the clock source of the divider that generates the clock to the LFAST. 00 Selects the crystal oscillator (XOSC). 01 Selects the SIPI_REFCLK pin. 10 Selects the peripheral clock coming from the PLL. 11 Reserved.
20–24 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
25–31 LFDIV	LFAST Clock Divider. Selects the divider value for the clock that feeds the LFAST module, according to the following formula: DivisionFactor = LFDIV + 1

8.2.48 PSI5 Clock Configuration Register (SIU_PSCLKCFG)

The SIU_PSCLKCFG register defines the clock division factors to generate the clocks to the PSI5 module. [Figure 19-1](#) shows the PSDIV and PSDIV1M fields in the clock generation scheme.

Address: 0h base + 9CCh offset = 9CCh

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	LCK	0							PSDIV							
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0							PSDIV1M								
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

SIU_PSCLKCFG field descriptions

Field	Description
0 LCK	Write Lock Bit. When the LCK is written with a value of 1, writes to all other bits and fields in the SIU_PSCLKCFG register are disabled. The LCK bit can be set/cleared by software at any time. 0 SIU_PSCLKCFG register is unlocked and writes are permitted. 1 SIU_PSCLKCFG register is locked, and only writes to the LCK bit are permitted.
1–7 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
8–15 PSDIV	PSI5 Clock Divider. Selects the divider value for the clock that feeds the PSI5 module, according to the following formula: DivisionFactor = PSDIV + 1
16–21 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
22–31 PSDIV1M	PSI5 Clock Divider 1MHz. Selects the divider value for the clock that feeds the 1MHz input of the PSI5 module, according to the following formula: DivisionFactor = PSDIV1M + 1

8.2.49 Reset Configuration Register (SIU_RCR)

SIU_RCR contains a field that configures reset operation.

Address: 0h base + 9D0h offset = 9D0h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0															RET																
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0*	0*	0*	0*

* Notes:

- RET field: RET is loaded with 1000b when the REE bit of the UTEST Reset Escalation Configuration DCF client is 1.

SIU_RCR field descriptions

Field	Description
0–27 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
28–31 RET	Reset Escalation Threshold. This field defines the number of reset cycles not caused by POR that the device can execute before being held permanently in reset state, waiting for a POR. The defined number of non-POR reset cycles is the decimal value of RET plus 1, except when RET is 0. A value of 0 in this field disables the reset escalation feature. Any CPU write to the RET field re-initiates the escalation counting process.

8.2.50 Lock-Step Status Register (SIU_LOCKSTEP)

The SIU_LOCKSTEP register contains a status bit that indicates whether the checker core is enabled or not. The checker core is enabled by setting the LOCKSTEP_EN in the Miscellaneous DCF client. A reset cycle is required for a change in LOCKSTEP_EN to take effect.

Address: 0h base + 9D4h offset = 9D4h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0															LSE
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

SIU_LOCKSTEP field descriptions

Field	Description
0–30 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
31 LSE	Lock Step Enable. Indicates whether the checker core is enabled or not. 0 Lock step operation is disabled. 1 Lock step operation is enabled.

8.2.51 Progressive Clock Switch Interrupt Enable Register (SIU_PCSE)

The SIU_PCSE controls the generation of an interrupt request when the SIU_PCSIFR[PCSI] becomes set at the end of a progressive clock switch (PCS) operation.

Address: 0h base + 9D8h offset = 9D8h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15															
R	0																														
W																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0															
Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31															
R	0																												PCSE		
W																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0															

SIU_PCSE field descriptions

Field	Description
0–30 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
31 PCSE	PCS Interrupt Enable. Enables or disables the generation of an interrupt request when the SIU_PCSIFR[PCSI] becomes set at the end of a progressive clock switch operation. 0 Interrupt disabled. 1 Interrupt enabled.

8.2.52 Progressive Clock Switch Interrupt Flag Register (SIU_PCSIFR)

The SIU_PCSIFR contains an interrupt flag that is set when the progressive clock switch operation finishes.

Address: 0h base + 9DCh offset = 9DCh

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0														PCSMS	PCSI
W																w1c
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

SIU_PCSIFR field descriptions

Field	Description
0–28 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
29–30 PCSMS	PCS Status. Indicates the clock source which effectively drives the system clock. Note that the clock source indicated by PCSMS may be different than the one requested by SIU_SYSDIV[SYSCLKSEL] because the PCS may still be active, or because the PCS may have failed to switch to the new clock (for example, if it was programmed incorrectly). After the PCSI flag gets set, indicating that the PCS operation is finished, software can read the value of the PCSMS field to find out if the PCS switch was successful. 00 The system clock is driven by the 16 MHz Internal RC Oscillator (IRC). 01 The system clock is driven by the external oscillator (XOSC). 10 The system clock is driven by the FM PLL (PLL1). 11 The system clock is driven by the non-FM PLL (PLL0).
31 PCSI	PCS Interrupt Flag. Indicates the end of a progressive clock switch operation. Must be cleared by writing '1' to the bit. 0 No PCS operation finished since last time this flag was cleared. 1 PCS operation finished.

8.2.53 Fast Ethernet Controller Control Register (SIU_FECCR)

The SIU_FECCR is used to select between the Media Independent Interface Lite (MII-Lite) and the Reduced Media Independent Interface (RMII) modes of the Fast Ethernet Controller (FEC).

Address: 0h base + 9E0h offset = 9E0h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0															FM
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

SIU_FECCR field descriptions

Field	Description
0–30 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
31 FM	FEC Mode. Selects between MII-Lite and RMII modes in the FEC module. Note that this bit does not affect the configuration of pads associated with the FEC operation. Pad configuration must be done in the PCR registers. 0 RMII mode is selected. 1 MII-Lite mode is selected.

8.2.54 ECC Error Injection Register (SIU_ECCEIR)

The SIU_ECCEIR is used to inject ECC errors into reading operations on the MCAN RAM array. This register is intended to support testing of the error reporting path from the MCAN ECC logic to the FCCU input. Before programming this register, it is recommended to disable MCAN operations and then access the MCAN memories directly with the CPU.

ECC errors will be introduced into the read data path, therefore the memory contents remain intact. As soon as the error injection is disabled, normal MCAN operation can be resumed without the need to fix any memory content.

Address: 0h base + 9E4h offset = 9E4h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0								INVC		CHKINVM					
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

SIU_ECCEIR field descriptions

Field	Description
0–23 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
24 INVC	Invert Control. Controls whether the ECC checkbits can be inverted during memory read operations on the MCAN memory array. 0 No checkbit substitution is performed on MCAN memory read accesses. 1 Checkbit substitution is performed on MCAN write operations onto the MCAN memory.
25–31 CHKINVM	Check Bit Invert Mask. This field specifies which bits of the ECC checkbits are inverted during read operations on the MCAN memory array. For each bit of this field: 0 The bit is not inverted. 1 The bit is inverted.

8.2.55 Parallel GPIO Pin Data Output Registers (SIU_PGPDO_n)

The PGPDO_x registers are written to by software to drive data out on external GPIO pins. These registers access the same GPIO pins accessed by the SIU_GPDO0 - SIU_GPDO511 registers, and they share the same hardware registers. For example, SIU_PGPDO0 bit 0 is the same as SIU_GPDO0 bit 7, SIU_PGPDO0 bit 1 is the same as SIU_GPDO1 bit 7, and so on.

The parallel GPIO read/write addresses map to the same physical addresses of the normal GPIO. This way both GPDO and corresponding PGPDO registers will be updated by a single write to either registers.

Address: 0h base + C00h offset + (4d × i), where i=0d to 15d

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	PGPDO0	PGPDO1	PGPDO2	PGPDO3	PGPDO4	PGPDO5	PGPDO6	PGPDO7	PGPDO8	PGPDO9	PGPDO10	PGPDO11	PGPDO12	PGPDO13	PGPDO14	PGPDO15
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Memory Map and Register Definition

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	PGPDO16	PGPDO17	PGPDO18	PGPDO19	PGPDO20	PGPDO21	PGPDO22	PGPDO23	PGPDO24	PGPDO25	PGPDO26	PGPDO27	PGPDO28	PGPDO29	PGPDO30	PGPDO31
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

SIU_PGPDO_n field descriptions

Field	Description
0 PGPDO0	Pin Data Out. This bit stores the data to be driven out on the external GPIO pin controlled by this register. 0 Logic low value is driven on the GPIO pin when the pin is configured as an output. 1 Logic high value is driven on the GPIO pin when the pin is configured as an output.
1 PGPDO1	Pin Data Out. This bit stores the data to be driven out on the external GPIO pin controlled by this register. 0 Logic low value is driven on the GPIO pin when the pin is configured as an output. 1 Logic high value is driven on the GPIO pin when the pin is configured as an output.
2 PGPDO2	Pin Data Out. This bit stores the data to be driven out on the external GPIO pin controlled by this register. 0 Logic low value is driven on the GPIO pin when the pin is configured as an output. 1 Logic high value is driven on the GPIO pin when the pin is configured as an output.
3 PGPDO3	Pin Data Out. This bit stores the data to be driven out on the external GPIO pin controlled by this register. 0 Logic low value is driven on the GPIO pin when the pin is configured as an output. 1 Logic high value is driven on the GPIO pin when the pin is configured as an output.
4 PGPDO4	Pin Data Out. This bit stores the data to be driven out on the external GPIO pin controlled by this register. 0 Logic low value is driven on the GPIO pin when the pin is configured as an output. 1 Logic high value is driven on the GPIO pin when the pin is configured as an output.
5 PGPDO5	Pin Data Out. This bit stores the data to be driven out on the external GPIO pin controlled by this register. 0 Logic low value is driven on the GPIO pin when the pin is configured as an output. 1 Logic high value is driven on the GPIO pin when the pin is configured as an output.
6 PGPDO6	Pin Data Out. This bit stores the data to be driven out on the external GPIO pin controlled by this register. 0 Logic low value is driven on the GPIO pin when the pin is configured as an output. 1 Logic high value is driven on the GPIO pin when the pin is configured as an output.
7 PGPDO7	Pin Data Out. This bit stores the data to be driven out on the external GPIO pin controlled by this register. 0 Logic low value is driven on the GPIO pin when the pin is configured as an output. 1 Logic high value is driven on the GPIO pin when the pin is configured as an output.
8 PGPDO8	Pin Data Out. This bit stores the data to be driven out on the external GPIO pin controlled by this register. 0 Logic low value is driven on the GPIO pin when the pin is configured as an output. 1 Logic high value is driven on the GPIO pin when the pin is configured as an output.
9 PGPDO9	Pin Data Out. This bit stores the data to be driven out on the external GPIO pin controlled by this register. 0 Logic low value is driven on the GPIO pin when the pin is configured as an output. 1 Logic high value is driven on the GPIO pin when the pin is configured as an output.

Table continues on the next page...

SIU_PGPDOn field descriptions (continued)

Field	Description
10 PGPDO10	Pin Data Out. This bit stores the data to be driven out on the external GPIO pin controlled by this register. 0 Logic low value is driven on the GPIO pin when the pin is configured as an output. 1 Logic high value is driven on the GPIO pin when the pin is configured as an output.
11 PGPDO11	Pin Data Out. This bit stores the data to be driven out on the external GPIO pin controlled by this register. 0 Logic low value is driven on the GPIO pin when the pin is configured as an output. 1 Logic high value is driven on the GPIO pin when the pin is configured as an output.
12 PGPDO12	Pin Data Out. This bit stores the data to be driven out on the external GPIO pin controlled by this register. 0 Logic low value is driven on the GPIO pin when the pin is configured as an output. 1 Logic high value is driven on the GPIO pin when the pin is configured as an output.
13 PGPDO13	Pin Data Out. This bit stores the data to be driven out on the external GPIO pin controlled by this register. 0 Logic low value is driven on the GPIO pin when the pin is configured as an output. 1 Logic high value is driven on the GPIO pin when the pin is configured as an output.
14 PGPDO14	Pin Data Out. This bit stores the data to be driven out on the external GPIO pin controlled by this register. 0 Logic low value is driven on the GPIO pin when the pin is configured as an output. 1 Logic high value is driven on the GPIO pin when the pin is configured as an output.
15 PGPDO15	Pin Data Out. This bit stores the data to be driven out on the external GPIO pin controlled by this register. 0 Logic low value is driven on the GPIO pin when the pin is configured as an output. 1 Logic high value is driven on the GPIO pin when the pin is configured as an output.
16 PGPDO16	Pin Data Out. This bit stores the data to be driven out on the external GPIO pin controlled by this register. 0 Logic low value is driven on the GPIO pin when the pin is configured as an output. 1 Logic high value is driven on the GPIO pin when the pin is configured as an output.
17 PGPDO17	Pin Data Out. This bit stores the data to be driven out on the external GPIO pin controlled by this register. 0 Logic low value is driven on the GPIO pin when the pin is configured as an output. 1 Logic high value is driven on the GPIO pin when the pin is configured as an output.
18 PGPDO18	Pin Data Out. This bit stores the data to be driven out on the external GPIO pin controlled by this register. 0 Logic low value is driven on the GPIO pin when the pin is configured as an output. 1 Logic high value is driven on the GPIO pin when the pin is configured as an output.
19 PGPDO19	Pin Data Out. This bit stores the data to be driven out on the external GPIO pin controlled by this register. 0 Logic low value is driven on the GPIO pin when the pin is configured as an output. 1 Logic high value is driven on the GPIO pin when the pin is configured as an output.
20 PGPDO20	Pin Data Out. This bit stores the data to be driven out on the external GPIO pin controlled by this register. 0 Logic low value is driven on the GPIO pin when the pin is configured as an output. 1 Logic high value is driven on the GPIO pin when the pin is configured as an output.
21 PGPDO21	Pin Data Out. This bit stores the data to be driven out on the external GPIO pin controlled by this register. 0 Logic low value is driven on the GPIO pin when the pin is configured as an output. 1 Logic high value is driven on the GPIO pin when the pin is configured as an output.

Table continues on the next page...

SIU_PGPDOn field descriptions (continued)

Field	Description
22 PGPDO22	Pin Data Out. This bit stores the data to be driven out on the external GPIO pin controlled by this register. 0 Logic low value is driven on the GPIO pin when the pin is configured as an output. 1 Logic high value is driven on the GPIO pin when the pin is configured as an output.
23 PGPDO23	Pin Data Out. This bit stores the data to be driven out on the external GPIO pin controlled by this register. 0 Logic low value is driven on the GPIO pin when the pin is configured as an output. 1 Logic high value is driven on the GPIO pin when the pin is configured as an output.
24 PGPDO24	Pin Data Out. This bit stores the data to be driven out on the external GPIO pin controlled by this register. 0 Logic low value is driven on the GPIO pin when the pin is configured as an output. 1 Logic high value is driven on the GPIO pin when the pin is configured as an output.
25 PGPDO25	Pin Data Out. This bit stores the data to be driven out on the external GPIO pin controlled by this register. 0 Logic low value is driven on the GPIO pin when the pin is configured as an output. 1 Logic high value is driven on the GPIO pin when the pin is configured as an output.
26 PGPDO26	Pin Data Out. This bit stores the data to be driven out on the external GPIO pin controlled by this register. 0 Logic low value is driven on the GPIO pin when the pin is configured as an output. 1 Logic high value is driven on the GPIO pin when the pin is configured as an output.
27 PGPDO27	Pin Data Out. This bit stores the data to be driven out on the external GPIO pin controlled by this register. 0 Logic low value is driven on the GPIO pin when the pin is configured as an output. 1 Logic high value is driven on the GPIO pin when the pin is configured as an output.
28 PGPDO28	Pin Data Out. This bit stores the data to be driven out on the external GPIO pin controlled by this register. 0 Logic low value is driven on the GPIO pin when the pin is configured as an output. 1 Logic high value is driven on the GPIO pin when the pin is configured as an output.
29 PGPDO29	Pin Data Out. This bit stores the data to be driven out on the external GPIO pin controlled by this register. 0 Logic low value is driven on the GPIO pin when the pin is configured as an output. 1 Logic high value is driven on the GPIO pin when the pin is configured as an output.
30 PGPDO30	Pin Data Out. This bit stores the data to be driven out on the external GPIO pin controlled by this register. 0 Logic low value is driven on the GPIO pin when the pin is configured as an output. 1 Logic high value is driven on the GPIO pin when the pin is configured as an output.
31 PGPDO31	Pin Data Out. This bit stores the data to be driven out on the external GPIO pin controlled by this register. 0 Logic low value is driven on the GPIO pin when the pin is configured as an output. 1 Logic high value is driven on the GPIO pin when the pin is configured as an output.

8.2.56 Parallel GPIO Pin Data Input Registers (SIU_PGPDIn)

The GPDIX registers are read-only registers that allow reading of external GPIO pins. These registers access the same GPIO pins accessed by the SIU_GPDI0 - SIU_GPDI511 registers, and they share the same hardware registers. For example, SIU_PGPDIO bit 0 is the same as SIU_GPDI0 bit 7, SIU_PGPDIO bit 1 is the same as SIU_GPDI1 bit 7, and so on.

The parallel GPIO read/write addresses map to the same physical addresses of the normal GPIO. This way both the GPDI and corresponding PGPD registers are updated when a pin state changes, provided that the IBE bit is asserted in the corresponding PCR register.

Address: 0h base + C40h offset + (4d × i), where i=0d to 15d

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	PGPDIO	PGPD11	PGPD12	PGPD13	PGPD14	PGPD15	PGPD16	PGPD17	PGPD18	PGPD19	PGPD110	PGPD111	PGPD112	PGPD113	PGPD114	PGPD115
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	PGPD116	PGPD117	PGPD118	PGPD119	PGPD120	PGPD121	PGPD122	PGPD123	PGPD124	PGPD125	PGPD126	PGPD127	PGPD128	PGPD129	PGPD130	PGPD131
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

SIU_PGPDIn field descriptions

Field	Description
0 PGPD10	Pin Data In. This bit reflects the value of the external GPIO pin associated with the register, provided that the IBE bit is asserted in the corresponding PCR register. 0 The value of the GPIO pin is logic low. 1 The value of the GPIO pin is logic high.
1 PGPD11	Pin Data In. This bit reflects the value of the external GPIO pin associated with the register, provided that the IBE bit is asserted in the corresponding PCR register. 0 The value of the GPIO pin is logic low. 1 The value of the GPIO pin is logic high.
2 PGPD12	Pin Data In. This bit reflects the value of the external GPIO pin associated with the register, provided that the IBE bit is asserted in the corresponding PCR register. 0 The value of the GPIO pin is logic low. 1 The value of the GPIO pin is logic high.
3 PGPD13	Pin Data In. This bit reflects the value of the external GPIO pin associated with the register, provided that the IBE bit is asserted in the corresponding PCR register. 0 The value of the GPIO pin is logic low. 1 The value of the GPIO pin is logic high.
4 PGPD14	Pin Data In. This bit reflects the value of the external GPIO pin associated with the register, provided that the IBE bit is asserted in the corresponding PCR register. 0 The value of the GPIO pin is logic low. 1 The value of the GPIO pin is logic high.
5 PGPD15	Pin Data In. This bit reflects the value of the external GPIO pin associated with the register, provided that the IBE bit is asserted in the corresponding PCR register. 0 The value of the GPIO pin is logic low. 1 The value of the GPIO pin is logic high.
6 PGPD16	Pin Data In. This bit reflects the value of the external GPIO pin associated with the register, provided that the IBE bit is asserted in the corresponding PCR register. 0 The value of the GPIO pin is logic low. 1 The value of the GPIO pin is logic high.
7 PGPD17	Pin Data In. This bit reflects the value of the external GPIO pin associated with the register, provided that the IBE bit is asserted in the corresponding PCR register. 0 The value of the GPIO pin is logic low. 1 The value of the GPIO pin is logic high.
8 PGPD18	Pin Data In. This bit reflects the value of the external GPIO pin associated with the register, provided that the IBE bit is asserted in the corresponding PCR register. 0 The value of the GPIO pin is logic low. 1 The value of the GPIO pin is logic high.
9 PGPD19	Pin Data In. This bit reflects the value of the external GPIO pin associated with the register, provided that the IBE bit is asserted in the corresponding PCR register. 0 The value of the GPIO pin is logic low. 1 The value of the GPIO pin is logic high.

Table continues on the next page...

SIU_PGPDI_n field descriptions (continued)

Field	Description
10 PGPDI10	Pin Data In. This bit reflects the value of the external GPIO pin associated with the register, provided that the IBE bit is asserted in the corresponding PCR register. 0 The value of the GPIO pin is logic low. 1 The value of the GPIO pin is logic high.
11 PGPDI11	Pin Data In. This bit reflects the value of the external GPIO pin associated with the register, provided that the IBE bit is asserted in the corresponding PCR register. 0 The value of the GPIO pin is logic low. 1 The value of the GPIO pin is logic high.
12 PGPDI12	Pin Data In. This bit reflects the value of the external GPIO pin associated with the register, provided that the IBE bit is asserted in the corresponding PCR register. 0 The value of the GPIO pin is logic low. 1 The value of the GPIO pin is logic high.
13 PGPDI13	Pin Data In. This bit reflects the value of the external GPIO pin associated with the register, provided that the IBE bit is asserted in the corresponding PCR register. 0 The value of the GPIO pin is logic low. 1 The value of the GPIO pin is logic high.
14 PGPDI14	Pin Data In. This bit reflects the value of the external GPIO pin associated with the register, provided that the IBE bit is asserted in the corresponding PCR register. 0 The value of the GPIO pin is logic low. 1 The value of the GPIO pin is logic high.
15 PGPDI15	Pin Data In. This bit reflects the value of the external GPIO pin associated with the register, provided that the IBE bit is asserted in the corresponding PCR register. 0 The value of the GPIO pin is logic low. 1 The value of the GPIO pin is logic high.
16 PGPDI16	Pin Data In. This bit reflects the value of the external GPIO pin associated with the register, provided that the IBE bit is asserted in the corresponding PCR register. 0 The value of the GPIO pin is logic low. 1 The value of the GPIO pin is logic high.
17 PGPDI17	Pin Data In. This bit reflects the value of the external GPIO pin associated with the register, provided that the IBE bit is asserted in the corresponding PCR register. 0 The value of the GPIO pin is logic low. 1 The value of the GPIO pin is logic high.
18 PGPDI18	Pin Data In. This bit reflects the value of the external GPIO pin associated with the register, provided that the IBE bit is asserted in the corresponding PCR register. 0 The value of the GPIO pin is logic low. 1 The value of the GPIO pin is logic high.
19 PGPDI19	Pin Data In. This bit reflects the value of the external GPIO pin associated with the register, provided that the IBE bit is asserted in the corresponding PCR register. 0 The value of the GPIO pin is logic low. 1 The value of the GPIO pin is logic high.

Table continues on the next page...

SIU_PGPDI_n field descriptions (continued)

Field	Description
20 PGPDI20	Pin Data In. This bit reflects the value of the external GPIO pin associated with the register, provided that the IBE bit is asserted in the corresponding PCR register. 0 The value of the GPIO pin is logic low. 1 The value of the GPIO pin is logic high.
21 PGPDI21	Pin Data In. This bit reflects the value of the external GPIO pin associated with the register, provided that the IBE bit is asserted in the corresponding PCR register. 0 The value of the GPIO pin is logic low. 1 The value of the GPIO pin is logic high.
22 PGPDI22	Pin Data In. This bit reflects the value of the external GPIO pin associated with the register, provided that the IBE bit is asserted in the corresponding PCR register. 0 The value of the GPIO pin is logic low. 1 The value of the GPIO pin is logic high.
23 PGPDI23	Pin Data In. This bit reflects the value of the external GPIO pin associated with the register, provided that the IBE bit is asserted in the corresponding PCR register. 0 The value of the GPIO pin is logic low. 1 The value of the GPIO pin is logic high.
24 PGPDI24	Pin Data In. This bit reflects the value of the external GPIO pin associated with the register, provided that the IBE bit is asserted in the corresponding PCR register. 0 The value of the GPIO pin is logic low. 1 The value of the GPIO pin is logic high.
25 PGPDI25	Pin Data In. This bit reflects the value of the external GPIO pin associated with the register, provided that the IBE bit is asserted in the corresponding PCR register. 0 The value of the GPIO pin is logic low. 1 The value of the GPIO pin is logic high.
26 PGPDI26	Pin Data In. This bit reflects the value of the external GPIO pin associated with the register, provided that the IBE bit is asserted in the corresponding PCR register. 0 The value of the GPIO pin is logic low. 1 The value of the GPIO pin is logic high.
27 PGPDI27	Pin Data In. This bit reflects the value of the external GPIO pin associated with the register, provided that the IBE bit is asserted in the corresponding PCR register. 0 The value of the GPIO pin is logic low. 1 The value of the GPIO pin is logic high.
28 PGPDI28	Pin Data In. This bit reflects the value of the external GPIO pin associated with the register, provided that the IBE bit is asserted in the corresponding PCR register. 0 The value of the GPIO pin is logic low. 1 The value of the GPIO pin is logic high.
29 PGPDI29	Pin Data In. This bit reflects the value of the external GPIO pin associated with the register, provided that the IBE bit is asserted in the corresponding PCR register. 0 The value of the GPIO pin is logic low. 1 The value of the GPIO pin is logic high.

Table continues on the next page...

SIU_PGPDIn field descriptions (continued)

Field	Description
30 PGPDI30	Pin Data In. This bit reflects the value of the external GPIO pin associated with the register, provided that the IBE bit is asserted in the corresponding PCR register. 0 The value of the GPIO pin is logic low. 1 The value of the GPIO pin is logic high.
31 PGPDI31	Pin Data In. This bit reflects the value of the external GPIO pin associated with the register, provided that the IBE bit is asserted in the corresponding PCR register. 0 The value of the GPIO pin is logic low. 1 The value of the GPIO pin is logic high.

8.2.57 Masked Parallel GPIO Pin Data Output Registers (SIU_MPGPDO_n)

The MPGPDOn are write-only registers that are used to drive data out on external GPIO pins. These registers access the same GPIO pins accessed by SIU_GPDO0 - SIU_GPDO511 bit registers. The 16 least significant bits in the SIU_MPGPDO share the same hardware registers used by SIU_GPDO. For example, SIU_MPGPDO_n bit 16 is the same as SIU_GPDO_n bit 7, SIU_MPGPDO0 bit 17 is the same as SIU_GPDO1 bit 7, and so on.

The mask bits qualify the write operation of the data bits. Writing to DATAn is only effective if the corresponding MASKn bit is written to logic one at the same time (same access to the register). The value that was written to SIU_MPGPDO cannot be read back by a read operation to the same register. It can, however, be accessed by reading the corresponding SIU_GPDO register.

Address: 0h base + C80h offset + (4d × i), where i=0d to 31d

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W	MASK0	MASK1	MASK2	MASK3	MASK4	MASK5	MASK6	MASK7	MASK8	MASK9	MASK10	MASK11	MASK12	MASK13	MASK14	MASK15
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Memory Map and Register Definition

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W	DATA0	DATA1	DATA2	DATA3	DATA4	DATA5	DATA6	DATA7	DATA8	DATA9	DATA10	DATA11	DATA12	DATA13	DATA14	DATA15
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

SIU_MPGPDO_n field descriptions

Field	Description
0 MASK0	Mask Bit. This bit controls the write access to the corresponding GPDO. 0 Previous value defined by GPDO is maintained. 1 Corresponding GPDO is written with value defined by DATA field.
1 MASK1	Mask Bit. This bit controls the write access to the corresponding GPDO. 0 Previous value defined by GPDO is maintained. 1 Corresponding GPDO is written with value defined by DATA field.
2 MASK2	Mask Bit. This bit controls the write access to the corresponding GPDO. 0 Previous value defined by GPDO is maintained. 1 Corresponding GPDO is written with value defined by DATA field.
3 MASK3	Mask Bit. This bit controls the write access to the corresponding GPDO. 0 Previous value defined by GPDO is maintained. 1 Corresponding GPDO is written with value defined by DATA field.
4 MASK4	Mask Bit. This bit controls the write access to the corresponding GPDO. 0 Previous value defined by GPDO is maintained. 1 Corresponding GPDO is written with value defined by DATA field.
5 MASK5	Mask Bit. This bit controls the write access to the corresponding GPDO. 0 Previous value defined by GPDO is maintained. 1 Corresponding GPDO is written with value defined by DATA field.
6 MASK6	Mask Bit. This bit controls the write access to the corresponding GPDO. 0 Previous value defined by GPDO is maintained. 1 Corresponding GPDO is written with value defined by DATA field.
7 MASK7	Mask Bit. This bit controls the write access to the corresponding GPDO. 0 Previous value defined by GPDO is maintained. 1 Corresponding GPDO is written with value defined by DATA field.
8 MASK8	Mask Bit. This bit controls the write access to the corresponding GPDO. 0 Previous value defined by GPDO is maintained. 1 Corresponding GPDO is written with value defined by DATA field.

Table continues on the next page...

SIU_MPGPDOn field descriptions (continued)

Field	Description
9 MASK9	Mask Bit. This bit controls the write access to the corresponding GPDO. 0 Previous value defined by GPDO is maintained. 1 Corresponding GPDO is written with value defined by DATA field.
10 MASK10	Mask Bit. This bit controls the write access to the corresponding GPDO. 0 Previous value defined by GPDO is maintained. 1 Corresponding GPDO is written with value defined by DATA field.
11 MASK11	Mask Bit. This bit controls the write access to the corresponding GPDO. 0 Previous value defined by GPDO is maintained. 1 Corresponding GPDO is written with value defined by DATA field.
12 MASK12	Mask Bit. This bit controls the write access to the corresponding GPDO. 0 Previous value defined by GPDO is maintained. 1 Corresponding GPDO is written with value defined by DATA field.
13 MASK13	Mask Bit. This bit controls the write access to the corresponding GPDO. 0 Previous value defined by GPDO is maintained. 1 Corresponding GPDO is written with value defined by DATA field.
14 MASK14	Mask Bit. This bit controls the write access to the corresponding GPDO. 0 Previous value defined by GPDO is maintained. 1 Corresponding GPDO is written with value defined by DATA field.
15 MASK15	Mask Bit. This bit controls the write access to the corresponding GPDO. 0 Previous value defined by GPDO is maintained. 1 Corresponding GPDO is written with value defined by DATA field.
16 DATA0	Pin Data Out. This bit stores the data to be driven out on the external GPIO pin controlled by this register. 0 Logic low is driven on the corresponding GPIO pin when the pin is configured as an output. 1 Logic high is driven on the corresponding GPIO pin when the pin is configured as an output.
17 DATA1	Pin Data Out. This bit stores the data to be driven out on the external GPIO pin controlled by this register. 0 Logic low is driven on the corresponding GPIO pin when the pin is configured as an output. 1 Logic high is driven on the corresponding GPIO pin when the pin is configured as an output.
18 DATA2	Pin Data Out. This bit stores the data to be driven out on the external GPIO pin controlled by this register. 0 Logic low is driven on the corresponding GPIO pin when the pin is configured as an output. 1 Logic high is driven on the corresponding GPIO pin when the pin is configured as an output.
19 DATA3	Pin Data Out. This bit stores the data to be driven out on the external GPIO pin controlled by this register. 0 Logic low is driven on the corresponding GPIO pin when the pin is configured as an output. 1 Logic high is driven on the corresponding GPIO pin when the pin is configured as an output.
20 DATA4	Pin Data Out. This bit stores the data to be driven out on the external GPIO pin controlled by this register. 0 Logic low is driven on the corresponding GPIO pin when the pin is configured as an output. 1 Logic high is driven on the corresponding GPIO pin when the pin is configured as an output.

Table continues on the next page...

SIU_MPGPDO_n field descriptions (continued)

Field	Description
21 DATA5	Pin Data Out. This bit stores the data to be driven out on the external GPIO pin controlled by this register. 0 Logic low is driven on the corresponding GPIO pin when the pin is configured as an output. 1 Logic high is driven on the corresponding GPIO pin when the pin is configured as an output.
22 DATA6	Pin Data Out. This bit stores the data to be driven out on the external GPIO pin controlled by this register. 0 Logic low is driven on the corresponding GPIO pin when the pin is configured as an output. 1 Logic high is driven on the corresponding GPIO pin when the pin is configured as an output.
23 DATA7	Pin Data Out. This bit stores the data to be driven out on the external GPIO pin controlled by this register. 0 Logic low is driven on the corresponding GPIO pin when the pin is configured as an output. 1 Logic high is driven on the corresponding GPIO pin when the pin is configured as an output.
24 DATA8	Pin Data Out. This bit stores the data to be driven out on the external GPIO pin controlled by this register. 0 Logic low is driven on the corresponding GPIO pin when the pin is configured as an output. 1 Logic high is driven on the corresponding GPIO pin when the pin is configured as an output.
25 DATA9	Pin Data Out. This bit stores the data to be driven out on the external GPIO pin controlled by this register. 0 Logic low is driven on the corresponding GPIO pin when the pin is configured as an output. 1 Logic high is driven on the corresponding GPIO pin when the pin is configured as an output.
26 DATA10	Pin Data Out. This bit stores the data to be driven out on the external GPIO pin controlled by this register. 0 Logic low is driven on the corresponding GPIO pin when the pin is configured as an output. 1 Logic high is driven on the corresponding GPIO pin when the pin is configured as an output.
27 DATA11	Pin Data Out. This bit stores the data to be driven out on the external GPIO pin controlled by this register. 0 Logic low is driven on the corresponding GPIO pin when the pin is configured as an output. 1 Logic high is driven on the corresponding GPIO pin when the pin is configured as an output.
28 DATA12	Pin Data Out. This bit stores the data to be driven out on the external GPIO pin controlled by this register. 0 Logic low is driven on the corresponding GPIO pin when the pin is configured as an output. 1 Logic high is driven on the corresponding GPIO pin when the pin is configured as an output.
29 DATA13	Pin Data Out. This bit stores the data to be driven out on the external GPIO pin controlled by this register. 0 Logic low is driven on the corresponding GPIO pin when the pin is configured as an output. 1 Logic high is driven on the corresponding GPIO pin when the pin is configured as an output.
30 DATA14	Pin Data Out. This bit stores the data to be driven out on the external GPIO pin controlled by this register. 0 Logic low is driven on the corresponding GPIO pin when the pin is configured as an output. 1 Logic high is driven on the corresponding GPIO pin when the pin is configured as an output.
31 DATA15	Pin Data Out. This bit stores the data to be driven out on the external GPIO pin controlled by this register. 0 Logic low is driven on the corresponding GPIO pin when the pin is configured as an output. 1 Logic high is driven on the corresponding GPIO pin when the pin is configured as an output.

8.2.58 Mask-Output High Register (SIU_DSPI_nH)

Each bit in the DSPI serialized output frame can contain a signal routed from the output of one of three on-chip sources. The sources on this device are the eTPU module, eMIOS module, or a software-updated GPO data register. The mapping between module or data register and the output frame for a single DSPI channel is performed by independent 32-bit registers, one for each module plus one for the data register.

These registers are one of the possible sources for the DSPI serialized output frame. The purpose of these registers is to allow any combination of bits in each half of the 32 bit serialized data frame to be updated with a single 32-bit write operation, while allowing other bits to maintain their previous state. This is accomplished by writing a 16 bit masked value coherently with an update value contained in a 16 bit output field, and only updating those bits in the output register for which the corresponding mask bit is set. The mask bits are write-only. A value written to a MASK bit only takes effect on the corresponding DATA bit on the same write cycle in which it is written. Reading a mask bit always returns zero.

Address: 0h base + D00h offset + (8d × i), where i=0d to 3d

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W	MASK0	MASK1	MASK2	MASK3	MASK4	MASK5	MASK6	MASK7	MASK8	MASK9	MASK10	MASK11	MASK12	MASK13	MASK14	MASK15
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R																
W	DATA0	DATA1	DATA2	DATA3	DATA4	DATA5	DATA6	DATA7	DATA8	DATA9	DATA10	DATA11	DATA12	DATA13	DATA14	DATA15
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

SIU_DSPI_nH field descriptions

Field	Description
0 MASK0	Mask bit. These write-only bits control the write access to the DSPI GPO bits. Reading these bits returns 0. 0 Previous DSPI GPO value is kept. 1 Corresponding DSPI GPO bit is written with value defined by DATA field.
1 MASK1	Mask bit. These write-only bits control the write access to the DSPI GPO bits. Reading these bits returns 0. 0 Previous DSPI GPO value is kept. 1 Corresponding DSPI GPO bit is written with value defined by DATA field.
2 MASK2	Mask bit. These write-only bits control the write access to the DSPI GPO bits. Reading these bits returns 0. 0 Previous DSPI GPO value is kept. 1 Corresponding DSPI GPO bit is written with value defined by DATA field.
3 MASK3	Mask bit. These write-only bits control the write access to the DSPI GPO bits. Reading these bits returns 0. 0 Previous DSPI GPO value is kept. 1 Corresponding DSPI GPO bit is written with value defined by DATA field.
4 MASK4	Mask bit. These write-only bits control the write access to the DSPI GPO bits. Reading these bits returns 0. 0 Previous DSPI GPO value is kept. 1 Corresponding DSPI GPO bit is written with value defined by DATA field.
5 MASK5	Mask bit. These write-only bits control the write access to the DSPI GPO bits. Reading these bits returns 0. 0 Previous DSPI GPO value is kept. 1 Corresponding DSPI GPO bit is written with value defined by DATA field.
6 MASK6	Mask bit. These write-only bits control the write access to the DSPI GPO bits. Reading these bits returns 0. 0 Previous DSPI GPO value is kept. 1 Corresponding DSPI GPO bit is written with value defined by DATA field.
7 MASK7	Mask bit. These write-only bits control the write access to the DSPI GPO bits. Reading these bits returns 0. 0 Previous DSPI GPO value is kept. 1 Corresponding DSPI GPO bit is written with value defined by DATA field.
8 MASK8	Mask bit. These write-only bits control the write access to the DSPI GPO bits. Reading these bits returns 0. 0 Previous DSPI GPO value is kept. 1 Corresponding DSPI GPO bit is written with value defined by DATA field.
9 MASK9	Mask bit. These write-only bits control the write access to the DSPI GPO bits. Reading these bits returns 0. 0 Previous DSPI GPO value is kept. 1 Corresponding DSPI GPO bit is written with value defined by DATA field.

Table continues on the next page...

SIU_DSPI_nH field descriptions (continued)

Field	Description
10 MASK10	Mask bit. These write-only bits control the write access to the DSPI GPO bits. Reading these bits returns 0. 0 Previous DSPI GPO value is kept. 1 Corresponding DSPI GPO bit is written with value defined by DATA field.
11 MASK11	Mask bit. These write-only bits control the write access to the DSPI GPO bits. Reading these bits returns 0. 0 Previous DSPI GPO value is kept. 1 Corresponding DSPI GPO bit is written with value defined by DATA field.
12 MASK12	Mask bit. These write-only bits control the write access to the DSPI GPO bits. Reading these bits returns 0. 0 Previous DSPI GPO value is kept. 1 Corresponding DSPI GPO bit is written with value defined by DATA field.
13 MASK13	Mask bit. These write-only bits control the write access to the DSPI GPO bits. Reading these bits returns 0. 0 Previous DSPI GPO value is kept. 1 Corresponding DSPI GPO bit is written with value defined by DATA field.
14 MASK14	Mask bit. These write-only bits control the write access to the DSPI GPO bits. Reading these bits returns 0. 0 Previous DSPI GPO value is kept. 1 Corresponding DSPI GPO bit is written with value defined by DATA field.
15 MASK15	Mask bit. These write-only bits control the write access to the DSPI GPO bits. Reading these bits returns 0. 0 Previous DSPI GPO value is kept. 1 Corresponding DSPI GPO bit is written with value defined by DATA field.
16 DATA0	Pin Data Out. Stores the data to be driven out on the GPO for DSPI output controlled by this register. 0 Logic low value is driven to the DSPI serialization interface when this output is selected in SIU_DSPIAHLA/SIU_DSPIBHLB/SIU_DSPICHLC/SIU_DSPIDHLD. 1 Logic high value is driven to the DSPI serialization interface when this output is selected in SIU_DSPIAHLA/SIU_DSPIBHLB/SIU_DSPICHLC/SIU_DSPIDHLD.
17 DATA1	Pin Data Out. Stores the data to be driven out on the GPO for DSPI output controlled by this register. 0 Logic low value is driven to the DSPI serialization interface when this output is selected in SIU_DSPIAHLA/SIU_DSPIBHLB/SIU_DSPICHLC/SIU_DSPIDHLD. 1 Logic high value is driven to the DSPI serialization interface when this output is selected in SIU_DSPIAHLA/SIU_DSPIBHLB/SIU_DSPICHLC/SIU_DSPIDHLD.
18 DATA2	Pin Data Out. Stores the data to be driven out on the GPO for DSPI output controlled by this register. 0 Logic low value is driven to the DSPI serialization interface when this output is selected in SIU_DSPIAHLA/SIU_DSPIBHLB/SIU_DSPICHLC/SIU_DSPIDHLD. 1 Logic high value is driven to the DSPI serialization interface when this output is selected in SIU_DSPIAHLA/SIU_DSPIBHLB/SIU_DSPICHLC/SIU_DSPIDHLD.
19 DATA3	Pin Data Out. Stores the data to be driven out on the GPO for DSPI output controlled by this register.

Table continues on the next page...

SIU_DSPIAH field descriptions (continued)

Field	Description
	0 Logic low value is driven to the DSPI serialization interface when this output is selected in SIU_DSPIAHLA/SIU_DSPIBHLB/SIU_DSPICHLC/SIU_DSPIDHLD. 1 Logic high value is driven to the DSPI serialization interface when this output is selected in SIU_DSPIAHLA/SIU_DSPIBHLB/SIU_DSPICHLC/SIU_DSPIDHLD.
20 DATA4	Pin Data Out. Stores the data to be driven out on the GPO for DSPI output controlled by this register. 0 Logic low value is driven to the DSPI serialization interface when this output is selected in SIU_DSPIAHLA/SIU_DSPIBHLB/SIU_DSPICHLC/SIU_DSPIDHLD. 1 Logic high value is driven to the DSPI serialization interface when this output is selected in SIU_DSPIAHLA/SIU_DSPIBHLB/SIU_DSPICHLC/SIU_DSPIDHLD.
21 DATA5	Pin Data Out. Stores the data to be driven out on the GPO for DSPI output controlled by this register. 0 Logic low value is driven to the DSPI serialization interface when this output is selected in SIU_DSPIAHLA/SIU_DSPIBHLB/SIU_DSPICHLC/SIU_DSPIDHLD. 1 Logic high value is driven to the DSPI serialization interface when this output is selected in SIU_DSPIAHLA/SIU_DSPIBHLB/SIU_DSPICHLC/SIU_DSPIDHLD.
22 DATA6	Pin Data Out. Stores the data to be driven out on the GPO for DSPI output controlled by this register. 0 Logic low value is driven to the DSPI serialization interface when this output is selected in SIU_DSPIAHLA/SIU_DSPIBHLB/SIU_DSPICHLC/SIU_DSPIDHLD. 1 Logic high value is driven to the DSPI serialization interface when this output is selected in SIU_DSPIAHLA/SIU_DSPIBHLB/SIU_DSPICHLC/SIU_DSPIDHLD.
23 DATA7	Pin Data Out. Stores the data to be driven out on the GPO for DSPI output controlled by this register. 0 Logic low value is driven to the DSPI serialization interface when this output is selected in SIU_DSPIAHLA/SIU_DSPIBHLB/SIU_DSPICHLC/SIU_DSPIDHLD. 1 Logic high value is driven to the DSPI serialization interface when this output is selected in SIU_DSPIAHLA/SIU_DSPIBHLB/SIU_DSPICHLC/SIU_DSPIDHLD.
24 DATA8	Pin Data Out. Stores the data to be driven out on the GPO for DSPI output controlled by this register. 0 Logic low value is driven to the DSPI serialization interface when this output is selected in SIU_DSPIAHLA/SIU_DSPIBHLB/SIU_DSPICHLC/SIU_DSPIDHLD. 1 Logic high value is driven to the DSPI serialization interface when this output is selected in SIU_DSPIAHLA/SIU_DSPIBHLB/SIU_DSPICHLC/SIU_DSPIDHLD.
25 DATA9	Pin Data Out. Stores the data to be driven out on the GPO for DSPI output controlled by this register. 0 Logic low value is driven to the DSPI serialization interface when this output is selected in SIU_DSPIAHLA/SIU_DSPIBHLB/SIU_DSPICHLC/SIU_DSPIDHLD. 1 Logic high value is driven to the DSPI serialization interface when this output is selected in SIU_DSPIAHLA/SIU_DSPIBHLB/SIU_DSPICHLC/SIU_DSPIDHLD.
26 DATA10	Pin Data Out. Stores the data to be driven out on the GPO for DSPI output controlled by this register. 0 Logic low value is driven to the DSPI serialization interface when this output is selected in SIU_DSPIAHLA/SIU_DSPIBHLB/SIU_DSPICHLC/SIU_DSPIDHLD. 1 Logic high value is driven to the DSPI serialization interface when this output is selected in SIU_DSPIAHLA/SIU_DSPIBHLB/SIU_DSPICHLC/SIU_DSPIDHLD.
27 DATA11	Pin Data Out. Stores the data to be driven out on the GPO for DSPI output controlled by this register.

Table continues on the next page...

SIU_DSPIInH field descriptions (continued)

Field	Description
	0 Logic low value is driven to the DSPI serialization interface when this output is selected in SIU_DSPIAHLA/SIU_DSPIBHLB/SIU_DSPICHLC/SIU_DSPIDHLD. 1 Logic high value is driven to the DSPI serialization interface when this output is selected in SIU_DSPIAHLA/SIU_DSPIBHLB/SIU_DSPICHLC/SIU_DSPIDHLD.
28 DATA12	Pin Data Out. Stores the data to be driven out on the GPO for DSPI output controlled by this register. 0 Logic low value is driven to the DSPI serialization interface when this output is selected in SIU_DSPIAHLA/SIU_DSPIBHLB/SIU_DSPICHLC/SIU_DSPIDHLD. 1 Logic high value is driven to the DSPI serialization interface when this output is selected in SIU_DSPIAHLA/SIU_DSPIBHLB/SIU_DSPICHLC/SIU_DSPIDHLD.
29 DATA13	Pin Data Out. Stores the data to be driven out on the GPO for DSPI output controlled by this register. 0 Logic low value is driven to the DSPI serialization interface when this output is selected in SIU_DSPIAHLA/SIU_DSPIBHLB/SIU_DSPICHLC/SIU_DSPIDHLD. 1 Logic high value is driven to the DSPI serialization interface when this output is selected in SIU_DSPIAHLA/SIU_DSPIBHLB/SIU_DSPICHLC/SIU_DSPIDHLD.
30 DATA14	Pin Data Out. Stores the data to be driven out on the GPO for DSPI output controlled by this register. 0 Logic low value is driven to the DSPI serialization interface when this output is selected in SIU_DSPIAHLA/SIU_DSPIBHLB/SIU_DSPICHLC/SIU_DSPIDHLD. 1 Logic high value is driven to the DSPI serialization interface when this output is selected in SIU_DSPIAHLA/SIU_DSPIBHLB/SIU_DSPICHLC/SIU_DSPIDHLD.
31 DATA15	Pin Data Out. Stores the data to be driven out on the GPO for DSPI output controlled by this register. 0 Logic low value is driven to the DSPI serialization interface when this output is selected in SIU_DSPIAHLA/SIU_DSPIBHLB/SIU_DSPICHLC/SIU_DSPIDHLD. 1 Logic high value is driven to the DSPI serialization interface when this output is selected in SIU_DSPIAHLA/SIU_DSPIBHLB/SIU_DSPICHLC/SIU_DSPIDHLD.

8.2.59 Mask-Output Low Register (SIU_DSPIInL)

Each bit in the DSPI serialized output frame can contain a signal routed from the output of one of three on-chip sources. The sources on this device are the eTPU module, eMIOS module, or a software-updated GPO data register. The mapping between module or data register and the output frame for a single DSPI channel is performed by independent 32-bit registers, one for each module plus one for the data register.

These registers are one of the possible sources for the DSPI serialized output frame. The purpose of these registers is to allow any combination of bits in each half of the 32 bit serialized data frame to be updated with a single 32-bit write operation, while allowing other bits to maintain their previous state. This is accomplished by writing a 16 bit masked value coherently with an update value contained in a 16 bit output field, and only updating those bits in the output register for which the corresponding mask bit is set. The

Memory Map and Register Definition

mask bits are write-only. A value written to a MASK bit only takes effect on the corresponding DATA bit on the same write cycle in which it is written. Reading a mask bit always returns zero.

Address: 0h base + D04h offset + (8d × i), where i=0d to 3d

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W	MASK16	MASK17	MASK18	MASK19	MASK20	MASK21	MASK22	MASK23	MASK24	MASK25	MASK26	MASK27	MASK28	MASK29	MASK30	MASK31
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	DATA16	DATA17	DATA18	DATA19	DATA20	DATA21	DATA22	DATA23	DATA24	DATA25	DATA26	DATA27	DATA28	DATA29	DATA30	DATA31
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

SIU_DSPI_nL field descriptions

Field	Description
0 MASK16	Mask bit. These write-only bits control the write access to the DSPI GPO bits. Reading these bits returns 0. 0 Previous DSPI GPO value is kept. 1 Corresponding DSPI GPO bit is written with value defined by DATA field.
1 MASK17	Mask bit. These write-only bits control the write access to the DSPI GPO bits. Reading these bits returns 0. 0 Previous DSPI GPO value is kept. 1 Corresponding DSPI GPO bit is written with value defined by DATA field.
2 MASK18	Mask bit. These write-only bits control the write access to the DSPI GPO bits. Reading these bits returns 0. 0 Previous DSPI GPO value is kept. 1 Corresponding DSPI GPO bit is written with value defined by DATA field.
3 MASK19	Mask bit. These write-only bits control the write access to the DSPI GPO bits. Reading these bits returns 0. 0 Previous DSPI GPO value is kept. 1 Corresponding DSPI GPO bit is written with value defined by DATA field.

Table continues on the next page...

SIU_DSPI_nL field descriptions (continued)

Field	Description
4 MASK20	Mask bit. These write-only bits control the write access to the DSPI GPO bits. Reading these bits returns 0. 0 Previous DSPI GPO value is kept. 1 Corresponding DSPI GPO bit is written with value defined by DATA field.
5 MASK21	Mask bit. These write-only bits control the write access to the DSPI GPO bits. Reading these bits returns 0. 0 Previous DSPI GPO value is kept. 1 Corresponding DSPI GPO bit is written with value defined by DATA field.
6 MASK22	Mask bit. These write-only bits control the write access to the DSPI GPO bits. Reading these bits returns 0. 0 Previous DSPI GPO value is kept. 1 Corresponding DSPI GPO bit is written with value defined by DATA field.
7 MASK23	Mask bit. These write-only bits control the write access to the DSPI GPO bits. Reading these bits returns 0. 0 Previous DSPI GPO value is kept. 1 Corresponding DSPI GPO bit is written with value defined by DATA field.
8 MASK24	Mask bit. These write-only bits control the write access to the DSPI GPO bits. Reading these bits returns 0. 0 Previous DSPI GPO value is kept. 1 Corresponding DSPI GPO bit is written with value defined by DATA field.
9 MASK25	Mask bit. These write-only bits control the write access to the DSPI GPO bits. Reading these bits returns 0. 0 Previous DSPI GPO value is kept. 1 Corresponding DSPI GPO bit is written with value defined by DATA field.
10 MASK26	Mask bit. These write-only bits control the write access to the DSPI GPO bits. Reading these bits returns 0. 0 Previous DSPI GPO value is kept. 1 Corresponding DSPI GPO bit is written with value defined by DATA field.
11 MASK27	Mask bit. These write-only bits control the write access to the DSPI GPO bits. Reading these bits returns 0. 0 Previous DSPI GPO value is kept. 1 Corresponding DSPI GPO bit is written with value defined by DATA field.
12 MASK28	Mask bit. These write-only bits control the write access to the DSPI GPO bits. Reading these bits returns 0. 0 Previous DSPI GPO value is kept. 1 Corresponding DSPI GPO bit is written with value defined by DATA field.
13 MASK29	Mask bit. These write-only bits control the write access to the DSPI GPO bits. Reading these bits returns 0. 0 Previous DSPI GPO value is kept. 1 Corresponding DSPI GPO bit is written with value defined by DATA field.

Table continues on the next page...

SIU_DSPInL field descriptions (continued)

Field	Description
14 MASK30	Mask bit. These write-only bits control the write access to the DSPI GPO bits. Reading these bits returns 0. 0 Previous DSPI GPO value is kept. 1 Corresponding DSPI GPO bit is written with value defined by DATA field.
15 MASK31	Mask bit. These write-only bits control the write access to the DSPI GPO bits. Reading these bits returns 0. 0 Previous DSPI GPO value is kept. 1 Corresponding DSPI GPO bit is written with value defined by DATA field.
16 DATA16	Pin Data Out. Stores the data to be driven out on the GPO for DSPI output controlled by this register. 0 Logic low value is driven to the DSPI serialization interface when this output is selected in SIU_DSPIAHLA/SIU_DSPIBHLB/SIU_DSPICHLC/SIU_DSPIDHLD. 1 Logic high value is driven to the DSPI serialization interface when this output is selected in SIU_DSPIAHLA/SIU_DSPIBHLB/SIU_DSPICHLC/SIU_DSPIDHLD.
17 DATA17	Pin Data Out. Stores the data to be driven out on the GPO for DSPI output controlled by this register. 0 Logic low value is driven to the DSPI serialization interface when this output is selected in SIU_DSPIAHLA/SIU_DSPIBHLB/SIU_DSPICHLC/SIU_DSPIDHLD. 1 Logic high value is driven to the DSPI serialization interface when this output is selected in SIU_DSPIAHLA/SIU_DSPIBHLB/SIU_DSPICHLC/SIU_DSPIDHLD.
18 DATA18	Pin Data Out. Stores the data to be driven out on the GPO for DSPI output controlled by this register. 0 Logic low value is driven to the DSPI serialization interface when this output is selected in SIU_DSPIAHLA/SIU_DSPIBHLB/SIU_DSPICHLC/SIU_DSPIDHLD. 1 Logic high value is driven to the DSPI serialization interface when this output is selected in SIU_DSPIAHLA/SIU_DSPIBHLB/SIU_DSPICHLC/SIU_DSPIDHLD.
19 DATA19	Pin Data Out. Stores the data to be driven out on the GPO for DSPI output controlled by this register. 0 Logic low value is driven to the DSPI serialization interface when this output is selected in SIU_DSPIAHLA/SIU_DSPIBHLB/SIU_DSPICHLC/SIU_DSPIDHLD. 1 Logic high value is driven to the DSPI serialization interface when this output is selected in SIU_DSPIAHLA/SIU_DSPIBHLB/SIU_DSPICHLC/SIU_DSPIDHLD.
20 DATA20	Pin Data Out. Stores the data to be driven out on the GPO for DSPI output controlled by this register. 0 Logic low value is driven to the DSPI serialization interface when this output is selected in SIU_DSPIAHLA/SIU_DSPIBHLB/SIU_DSPICHLC/SIU_DSPIDHLD. 1 Logic high value is driven to the DSPI serialization interface when this output is selected in SIU_DSPIAHLA/SIU_DSPIBHLB/SIU_DSPICHLC/SIU_DSPIDHLD.
21 DATA21	Pin Data Out. Stores the data to be driven out on the GPO for DSPI output controlled by this register. 0 Logic low value is driven to the DSPI serialization interface when this output is selected in SIU_DSPIAHLA/SIU_DSPIBHLB/SIU_DSPICHLC/SIU_DSPIDHLD. 1 Logic high value is driven to the DSPI serialization interface when this output is selected in SIU_DSPIAHLA/SIU_DSPIBHLB/SIU_DSPICHLC/SIU_DSPIDHLD.
22 DATA22	Pin Data Out. Stores the data to be driven out on the GPO for DSPI output controlled by this register.

Table continues on the next page...

SIU_DSPInL field descriptions (continued)

Field	Description
	0 Logic low value is driven to the DSPI serialization interface when this output is selected in SIU_DSPIAHLA/SIU_DSPIBHLB/SIU_DSPICHLC/SIU_DSPIDHLD. 1 Logic high value is driven to the DSPI serialization interface when this output is selected in SIU_DSPIAHLA/SIU_DSPIBHLB/SIU_DSPICHLC/SIU_DSPIDHLD.
23 DATA23	Pin Data Out. Stores the data to be driven out on the GPO for DSPI output controlled by this register. 0 Logic low value is driven to the DSPI serialization interface when this output is selected in SIU_DSPIAHLA/SIU_DSPIBHLB/SIU_DSPICHLC/SIU_DSPIDHLD. 1 Logic high value is driven to the DSPI serialization interface when this output is selected in SIU_DSPIAHLA/SIU_DSPIBHLB/SIU_DSPICHLC/SIU_DSPIDHLD.
24 DATA24	Pin Data Out. Stores the data to be driven out on the GPO for DSPI output controlled by this register. 0 Logic low value is driven to the DSPI serialization interface when this output is selected in SIU_DSPIAHLA/SIU_DSPIBHLB/SIU_DSPICHLC/SIU_DSPIDHLD. 1 Logic high value is driven to the DSPI serialization interface when this output is selected in SIU_DSPIAHLA/SIU_DSPIBHLB/SIU_DSPICHLC/SIU_DSPIDHLD.
25 DATA25	Pin Data Out. Stores the data to be driven out on the GPO for DSPI output controlled by this register. 0 Logic low value is driven to the DSPI serialization interface when this output is selected in SIU_DSPIAHLA/SIU_DSPIBHLB/SIU_DSPICHLC/SIU_DSPIDHLD. 1 Logic high value is driven to the DSPI serialization interface when this output is selected in SIU_DSPIAHLA/SIU_DSPIBHLB/SIU_DSPICHLC/SIU_DSPIDHLD.
26 DATA26	Pin Data Out. Stores the data to be driven out on the GPO for DSPI output controlled by this register. 0 Logic low value is driven to the DSPI serialization interface when this output is selected in SIU_DSPIAHLA/SIU_DSPIBHLB/SIU_DSPICHLC/SIU_DSPIDHLD. 1 Logic high value is driven to the DSPI serialization interface when this output is selected in SIU_DSPIAHLA/SIU_DSPIBHLB/SIU_DSPICHLC/SIU_DSPIDHLD.
27 DATA27	Pin Data Out. Stores the data to be driven out on the GPO for DSPI output controlled by this register. 0 Logic low value is driven to the DSPI serialization interface when this output is selected in SIU_DSPIAHLA/SIU_DSPIBHLB/SIU_DSPICHLC/SIU_DSPIDHLD. 1 Logic high value is driven to the DSPI serialization interface when this output is selected in SIU_DSPIAHLA/SIU_DSPIBHLB/SIU_DSPICHLC/SIU_DSPIDHLD.
28 DATA28	Pin Data Out. Stores the data to be driven out on the GPO for DSPI output controlled by this register. 0 Logic low value is driven to the DSPI serialization interface when this output is selected in SIU_DSPIAHLA/SIU_DSPIBHLB/SIU_DSPICHLC/SIU_DSPIDHLD. 1 Logic high value is driven to the DSPI serialization interface when this output is selected in SIU_DSPIAHLA/SIU_DSPIBHLB/SIU_DSPICHLC/SIU_DSPIDHLD.
29 DATA29	Pin Data Out. Stores the data to be driven out on the GPO for DSPI output controlled by this register. 0 Logic low value is driven to the DSPI serialization interface when this output is selected in SIU_DSPIAHLA/SIU_DSPIBHLB/SIU_DSPICHLC/SIU_DSPIDHLD. 1 Logic high value is driven to the DSPI serialization interface when this output is selected in SIU_DSPIAHLA/SIU_DSPIBHLB/SIU_DSPICHLC/SIU_DSPIDHLD.
30 DATA30	Pin Data Out. Stores the data to be driven out on the GPO for DSPI output controlled by this register.

Table continues on the next page...

SIU_DSPInL field descriptions (continued)

Field	Description
	0 Logic low value is driven to the DSPI serialization interface when this output is selected in SIU_DSPIAHLA/SIU_DSPIBHLB/SIU_DSPICHLC/SIU_DSPIDHLD.
	1 Logic high value is driven to the DSPI serialization interface when this output is selected in SIU_DSPIAHLA/SIU_DSPIBHLB/SIU_DSPICHLC/SIU_DSPIDHLD.
31 DATA31	Pin Data Out. Stores the data to be driven out on the GPO for DSPI output controlled by this register.
	0 Logic low value is driven to the DSPI serialization interface when this output is selected in SIU_DSPIAHLA/SIU_DSPIBHLB/SIU_DSPICHLC/SIU_DSPIDHLD.
	1 Logic high value is driven to the DSPI serialization interface when this output is selected in SIU_DSPIAHLA/SIU_DSPIBHLB/SIU_DSPICHLC/SIU_DSPIDHLD.

8.2.60 Serialized Output Signal Selection Register for DSPI_A (SIU_ETPUBA)

Each bit in the DSPI serialized output frame can contain a signal routed from the output of one of three on-chip sources. The sources on this device are the eTPU module, eMIOS module, or a software-updated GPO data register. The mapping between module or data register and the output frame for a single DSPI channel is performed by independent 32-bit registers, one for each module plus one for the data register.

The following register is used by DSPI_A to select the sources of the serialized output when running in DSI or CSI configuration.

Each register bit enables a path from the eTPU_B channel, eMIOS channel and data register bit SIU_DSPIAH/SIU_DSPIAL to the equivalent bit position in the DSPI_A serialized output frame. The user must ensure that bit selections from each of these registers do not overlap. Multiple sources are logically ORed, which provides the potential for combining outputs from multiple timer channels and data registers to produce more complex bit behavior.

Address: 0h base + D40h offset = D40h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	ETPUB15	ETPUB14	ETPUB13	ETPUB12	ETPUB11	ETPUB10	ETPUB9	ETPUB8	ETPUB7	ETPUB6	ETPUB5	ETPUB4	ETPUB3	ETPUB2	ETPUB1	ETPUB0
W	ETPUB15	ETPUB14	ETPUB13	ETPUB12	ETPUB11	ETPUB10	ETPUB9	ETPUB8	ETPUB7	ETPUB6	ETPUB5	ETPUB4	ETPUB3	ETPUB2	ETPUB1	ETPUB0
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	ETPUB31	ETPUB30	ETPUB29	ETPUB28	ETPUB27	ETPUB26	ETPUB25	ETPUB24	ETPUB23	ETPUB22	ETPUB21	ETPUB20	ETPUB19	ETPUB18	ETPUB17	ETPUB16
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

SIU_ETPUBA field descriptions

Field	Description
0 ETPUB15	eTPU_B Channel Select. Select an eTPU_B channel to source the serialized DSPI_A output. 0 This bit in the DSPI_A serialized output frame will not use the respective eTPU_B channel. 1 This bit in the DSPI_A serialized output frame will use the respective eTPU_B channel.
1 ETPUB14	eTPU_B Channel Select. Select an eTPU_B channel to source the serialized DSPI_A output. 0 This bit in the DSPI_A serialized output frame will not use the respective eTPU_B channel. 1 This bit in the DSPI_A serialized output frame will use the respective eTPU_B channel.
2 ETPUB13	eTPU_B Channel Select. Select an eTPU_B channel to source the serialized DSPI_A output. 0 This bit in the DSPI_A serialized output frame will not use the respective eTPU_B channel. 1 This bit in the DSPI_A serialized output frame will use the respective eTPU_B channel.
3 ETPUB12	eTPU_B Channel Select. Select an eTPU_B channel to source the serialized DSPI_A output. 0 This bit in the DSPI_A serialized output frame will not use the respective eTPU_B channel. 1 This bit in the DSPI_A serialized output frame will use the respective eTPU_B channel.
4 ETPUB11	eTPU_B Channel Select. Select an eTPU_B channel to source the serialized DSPI_A output. 0 This bit in the DSPI_A serialized output frame will not use the respective eTPU_B channel. 1 This bit in the DSPI_A serialized output frame will use the respective eTPU_B channel.
5 ETPUB10	eTPU_B Channel Select. Select an eTPU_B channel to source the serialized DSPI_A output. 0 This bit in the DSPI_A serialized output frame will not use the respective eTPU_B channel. 1 This bit in the DSPI_A serialized output frame will use the respective eTPU_B channel.
6 ETPUB9	eTPU_B Channel Select. Select an eTPU_B channel to source the serialized DSPI_A output. 0 This bit in the DSPI_A serialized output frame will not use the respective eTPU_B channel. 1 This bit in the DSPI_A serialized output frame will use the respective eTPU_B channel.
7 ETPUB8	eTPU_B Channel Select. Select an eTPU_B channel to source the serialized DSPI_A output. 0 This bit in the DSPI_A serialized output frame will not use the respective eTPU_B channel. 1 This bit in the DSPI_A serialized output frame will use the respective eTPU_B channel.
8 ETPUB7	eTPU_B Channel Select. Select an eTPU_B channel to source the serialized DSPI_A output. 0 This bit in the DSPI_A serialized output frame will not use the respective eTPU_B channel. 1 This bit in the DSPI_A serialized output frame will use the respective eTPU_B channel.
9 ETPUB6	eTPU_B Channel Select. Select an eTPU_B channel to source the serialized DSPI_A output. 0 This bit in the DSPI_A serialized output frame will not use the respective eTPU_B channel. 1 This bit in the DSPI_A serialized output frame will use the respective eTPU_B channel.

Table continues on the next page...

SIU_ETPUBA field descriptions (continued)

Field	Description
10 ETPUB5	eTPU_B Channel Select. Select an eTPU_B channel to source the serialized DSPI_A output. 0 This bit in the DSPI_A serialized output frame will not use the respective eTPU_B channel. 1 This bit in the DSPI_A serialized output frame will use the respective eTPU_B channel.
11 ETPUB4	eTPU_B Channel Select. Select an eTPU_B channel to source the serialized DSPI_A output. 0 This bit in the DSPI_A serialized output frame will not use the respective eTPU_B channel. 1 This bit in the DSPI_A serialized output frame will use the respective eTPU_B channel.
12 ETPUB3	eTPU_B Channel Select. Select an eTPU_B channel to source the serialized DSPI_A output. 0 This bit in the DSPI_A serialized output frame will not use the respective eTPU_B channel. 1 This bit in the DSPI_A serialized output frame will use the respective eTPU_B channel.
13 ETPUB2	eTPU_B Channel Select. Select an eTPU_B channel to source the serialized DSPI_A output. 0 This bit in the DSPI_A serialized output frame will not use the respective eTPU_B channel. 1 This bit in the DSPI_A serialized output frame will use the respective eTPU_B channel.
14 ETPUB1	eTPU_B Channel Select. Select an eTPU_B channel to source the serialized DSPI_A output. 0 This bit in the DSPI_A serialized output frame will not use the respective eTPU_B channel. 1 This bit in the DSPI_A serialized output frame will use the respective eTPU_B channel.
15 ETPUB0	eTPU_B Channel Select. Select an eTPU_B channel to source the serialized DSPI_A output. 0 This bit in the DSPI_A serialized output frame will not use the respective eTPU_B channel. 1 This bit in the DSPI_A serialized output frame will use the respective eTPU_B channel.
16 ETPUB31	eTPU_B Channel Select. Select an eTPU_B channel to source the serialized DSPI_A output. 0 This bit in the DSPI_A serialized output frame will not use the respective eTPU_B channel. 1 This bit in the DSPI_A serialized output frame will use the respective eTPU_B channel.
17 ETPUB30	eTPU_B Channel Select. Select an eTPU_B channel to source the serialized DSPI_A output. 0 This bit in the DSPI_A serialized output frame will not use the respective eTPU_B channel. 1 This bit in the DSPI_A serialized output frame will use the respective eTPU_B channel.
18 ETPUB29	eTPU_B Channel Select. Select an eTPU_B channel to source the serialized DSPI_A output. 0 This bit in the DSPI_A serialized output frame will not use the respective eTPU_B channel. 1 This bit in the DSPI_A serialized output frame will use the respective eTPU_B channel.
19 ETPUB28	eTPU_B Channel Select. Select an eTPU_B channel to source the serialized DSPI_A output. 0 This bit in the DSPI_A serialized output frame will not use the respective eTPU_B channel. 1 This bit in the DSPI_A serialized output frame will use the respective eTPU_B channel.
20 ETPUB27	eTPU_B Channel Select. Select an eTPU_B channel to source the serialized DSPI_A output. 0 This bit in the DSPI_A serialized output frame will not use the respective eTPU_B channel. 1 This bit in the DSPI_A serialized output frame will use the respective eTPU_B channel.
21 ETPUB26	eTPU_B Channel Select. Select an eTPU_B channel to source the serialized DSPI_A output. 0 This bit in the DSPI_A serialized output frame will not use the respective eTPU_B channel. 1 This bit in the DSPI_A serialized output frame will use the respective eTPU_B channel.

Table continues on the next page...

SIU_ETPUBA field descriptions (continued)

Field	Description
22 ETPUB25	eTPU_B Channel Select. Select an eTPU_B channel to source the serialized DSPI_A output. 0 This bit in the DSPI_A serialized output frame will not use the respective eTPU_B channel. 1 This bit in the DSPI_A serialized output frame will use the respective eTPU_B channel.
23 ETPUB24	eTPU_B Channel Select. Select an eTPU_B channel to source the serialized DSPI_A output. 0 This bit in the DSPI_A serialized output frame will not use the respective eTPU_B channel. 1 This bit in the DSPI_A serialized output frame will use the respective eTPU_B channel.
24 ETPUB23	eTPU_B Channel Select. Select an eTPU_B channel to source the serialized DSPI_A output. 0 This bit in the DSPI_A serialized output frame will not use the respective eTPU_B channel. 1 This bit in the DSPI_A serialized output frame will use the respective eTPU_B channel.
25 ETPUB22	eTPU_B Channel Select. Select an eTPU_B channel to source the serialized DSPI_A output. 0 This bit in the DSPI_A serialized output frame will not use the respective eTPU_B channel. 1 This bit in the DSPI_A serialized output frame will use the respective eTPU_B channel.
26 ETPUB21	eTPU_B Channel Select. Select an eTPU_B channel to source the serialized DSPI_A output. 0 This bit in the DSPI_A serialized output frame will not use the respective eTPU_B channel. 1 This bit in the DSPI_A serialized output frame will use the respective eTPU_B channel.
27 ETPUB20	eTPU_B Channel Select. Select an eTPU_B channel to source the serialized DSPI_A output. 0 This bit in the DSPI_A serialized output frame will not use the respective eTPU_B channel. 1 This bit in the DSPI_A serialized output frame will use the respective eTPU_B channel.
28 ETPUB19	eTPU_B Channel Select. Select an eTPU_B channel to source the serialized DSPI_A output. 0 This bit in the DSPI_A serialized output frame will not use the respective eTPU_B channel. 1 This bit in the DSPI_A serialized output frame will use the respective eTPU_B channel.
29 ETPUB18	eTPU_B Channel Select. Select an eTPU_B channel to source the serialized DSPI_A output. 0 This bit in the DSPI_A serialized output frame will not use the respective eTPU_B channel. 1 This bit in the DSPI_A serialized output frame will use the respective eTPU_B channel.
30 ETPUB17	eTPU_B Channel Select. Select an eTPU_B channel to source the serialized DSPI_A output. 0 This bit in the DSPI_A serialized output frame will not use the respective eTPU_B channel. 1 This bit in the DSPI_A serialized output frame will use the respective eTPU_B channel.
31 ETPUB16	eTPU_B Channel Select. Select an eTPU_B channel to source the serialized DSPI_A output. 0 This bit in the DSPI_A serialized output frame will not use the respective eTPU_B channel. 1 This bit in the DSPI_A serialized output frame will use the respective eTPU_B channel.

8.2.61 Serialized Output Signal Selection Register for DSPI_A (SIU_EMIOA)

Each bit in the DSPI serialized output frame can contain a signal routed from the output of one of three on-chip sources. The sources on this device are the eTPU module, eMIOS module, or a software-updated GPO data register. The mapping between module or data register and the output frame for a single DSPI channel is performed by independent 32-bit registers, one for each module plus one for the data register.

The following register is used by DSPI_A to select the sources of the serialized output when running in DSI or CSI configuration.

Each register bit enables a path from the eTPU_B channel, eMIOS channel and data register bit SIU_DSPIAH/SIU_DSPIAL to the equivalent bit position in the DSPI_A serialized output frame. The user must ensure that bit selections from each of these registers do not overlap. Multiple sources are logically ORed, which provides the potential for combining outputs from multiple timer channels and data registers to produce more complex bit behavior.

Address: 0h base + D44h offset = D44h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	EMIOS0_7_	EMIOS0_6_	EMIOS0_5_	EMIOS0_4_	EMIOS0_3_	EMIOS0_2_	EMIOS0_1_	EMIOS0_0_	EMIOS1_0_	EMIOS1_1_	EMIOS1_2_	EMIOS1_3_	EMIOS1_4_	EMIOS1_5_	EMIOS1_6_	EMIOS1_7_
W	OUT0	OUT1	OUT2	OUT3	OUT4	OUT5	OUT6	OUT7	OUT8	OUT9	OUT10	OUT11	OUT12	OUT13	OUT14	OUT15
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	EMIOS0_16_	EMIOS0_17_	EMIOS0_18_	EMIOS0_19_	EMIOS0_20_	EMIOS0_21_	EMIOS0_22_	EMIOS0_23_	EMIOS0_0_	EMIOS0_1_	EMIOS0_2_	EMIOS0_3_	EMIOS0_4_	EMIOS0_5_	EMIOS0_6_	EMIOS0_7_
W	OUT16	OUT17	OUT18	OUT19	OUT20	OUT21	OUT22	OUT23	OUT24	OUT25	OUT26	OUT27	OUT28	OUT29	OUT30	OUT31
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

SIU_EMIOA field descriptions

Field	Description
0 EMIOS0_7_ OUT0	eMIOS_0 Channel Select. Select an eMIOS_0 channel to source the serialized DSPI_A output. 0 This bit in the DSPI_A serialized output frame will not use eMIOS_0 channel 7. 1 This bit in the DSPI_A serialized output frame will use eMIOS_0 channel 7.

Table continues on the next page...

SIU_EMIOSA field descriptions (continued)

Field	Description
1 EMIOS0_6_ OUT1	eMIOS_0 Channel Select. Select an eMIOS_0 channel to source the serialized DSPI_A output. 0 This bit in the DSPI_A serialized output frame will not use eMIOS_0 channel 6. 1 This bit in the DSPI_A serialized output frame will use eMIOS_0 channel 6.
2 EMIOS0_5_ OUT2	eMIOS_0 Channel Select. Select an eMIOS_0 channel to source the serialized DSPI_A output. 0 This bit in the DSPI_A serialized output frame will not use eMIOS_0 channel 5. 1 This bit in the DSPI_A serialized output frame will use eMIOS_0 channel 5.
3 EMIOS0_4_ OUT3	eMIOS_0 Channel Select. Select an eMIOS_0 channel to source the serialized DSPI_A output. 0 This bit in the DSPI_A serialized output frame will not use eMIOS_0 channel 4. 1 This bit in the DSPI_A serialized output frame will use eMIOS_0 channel 4.
4 EMIOS0_3_ OUT4	eMIOS_0 Channel Select. Select an eMIOS_0 channel to source the serialized DSPI_A output. 0 This bit in the DSPI_A serialized output frame will not use eMIOS_0 channel 3. 1 This bit in the DSPI_A serialized output frame will use eMIOS_0 channel 3.
5 EMIOS0_2_ OUT5	eMIOS_0 Channel Select. Select an eMIOS_0 channel to source the serialized DSPI_A output. 0 This bit in the DSPI_A serialized output frame will not use eMIOS_0 channel 2. 1 This bit in the DSPI_A serialized output frame will use eMIOS_0 channel 2.
6 EMIOS0_1_ OUT6	eMIOS_0 Channel Select. Select an eMIOS_0 channel to source the serialized DSPI_A output. 0 This bit in the DSPI_A serialized output frame will not use eMIOS_0 channel 1. 1 This bit in the DSPI_A serialized output frame will use eMIOS_0 channel 1.
7 EMIOS0_0_ OUT7	eMIOS_0 Channel Select. Select an eMIOS_0 channel to source the serialized DSPI_A output. 0 This bit in the DSPI_A serialized output frame will not use eMIOS_0 channel 0. 1 This bit in the DSPI_A serialized output frame will use eMIOS_0 channel 0.
8 EMIOS1_0_ OUT8	eMIOS_1 Channel Select. Select an eMIOS_0 channel to source the serialized DSPI_A output. 0 This bit in the DSPI_A serialized output frame will not use eMIOS_1 channel 0. 1 This bit in the DSPI_A serialized output frame will use eMIOS_1 channel 0.
9 EMIOS1_1_ OUT9	eMIOS_1 Channel Select. Select an eMIOS_0 channel to source the serialized DSPI_A output. 0 This bit in the DSPI_A serialized output frame will not use eMIOS_1 channel 1. 1 This bit in the DSPI_A serialized output frame will use eMIOS_1 channel 1.
10 EMIOS1_2_ OUT10	eMIOS_1 Channel Select. Select an eMIOS_0 channel to source the serialized DSPI_A output. 0 This bit in the DSPI_A serialized output frame will not use eMIOS_1 channel 2. 1 This bit in the DSPI_A serialized output frame will use eMIOS_1 channel 2.
11 EMIOS1_3_ OUT11	eMIOS_1 Channel Select. Select an eMIOS_0 channel to source the serialized DSPI_A output. 0 This bit in the DSPI_A serialized output frame will not use eMIOS_1 channel 3. 1 This bit in the DSPI_A serialized output frame will use eMIOS_1 channel 3.
12 EMIOS1_4_ OUT12	eMIOS_1 Channel Select. Select an eMIOS_0 channel to source the serialized DSPI_A output. 0 This bit in the DSPI_A serialized output frame will not use eMIOS_1 channel 4. 1 This bit in the DSPI_A serialized output frame will use eMIOS_1 channel 4.

Table continues on the next page...

SIU_EMIOA field descriptions (continued)

Field	Description
13 EMIOS1_5_OUT13	eMIOS_1 Channel Select. Select an eMIOS_0 channel to source the serialized DSPI_A output. 0 This bit in the DSPI_A serialized output frame will not use eMIOS_1 channel 5. 1 This bit in the DSPI_A serialized output frame will use eMIOS_1 channel 5.
14 EMIOS1_6_OUT14	eMIOS_1 Channel Select. Select an eMIOS_0 channel to source the serialized DSPI_A output. 0 This bit in the DSPI_A serialized output frame will not use eMIOS_1 channel 6. 1 This bit in the DSPI_A serialized output frame will use eMIOS_1 channel 6.
15 EMIOS1_7_OUT15	eMIOS_1 Channel Select. Select an eMIOS_0 channel to source the serialized DSPI_A output. 0 This bit in the DSPI_A serialized output frame will not use eMIOS_1 channel 7. 1 This bit in the DSPI_A serialized output frame will use eMIOS_1 channel 7.
16 EMIOS0_16_OUT16	eMIOS_0 Channel Select. Select an eMIOS_0 channel to source the serialized DSPI_A output. 0 This bit in the DSPI_A serialized output frame will not use eMIOS_0 channel 16. 1 This bit in the DSPI_A serialized output frame will use eMIOS_0 channel 16.
17 EMIOS0_17_OUT17	eMIOS_0 Channel Select. Select an eMIOS_0 channel to source the serialized DSPI_A output. 0 This bit in the DSPI_A serialized output frame will not use eMIOS_0 channel 17. 1 This bit in the DSPI_A serialized output frame will use eMIOS_0 channel 17.
18 EMIOS0_18_OUT18	eMIOS_0 Channel Select. Select an eMIOS_0 channel to source the serialized DSPI_A output. 0 This bit in the DSPI_A serialized output frame will not use eMIOS_0 channel 18. 1 This bit in the DSPI_A serialized output frame will use eMIOS_0 channel 18.
19 EMIOS0_19_OUT19	eMIOS_0 Channel Select. Select an eMIOS_0 channel to source the serialized DSPI_A output. 0 This bit in the DSPI_A serialized output frame will not use eMIOS_0 channel 19. 1 This bit in the DSPI_A serialized output frame will use eMIOS_0 channel 19.
20 EMIOS0_20_OUT20	eMIOS_0 Channel Select. Select an eMIOS_0 channel to source the serialized DSPI_A output. 0 This bit in the DSPI_A serialized output frame will not use eMIOS_0 channel 20. 1 This bit in the DSPI_A serialized output frame will use eMIOS_0 channel 20.
21 EMIOS0_21_OUT21	eMIOS_0 Channel Select. Select an eMIOS_0 channel to source the serialized DSPI_A output. 0 This bit in the DSPI_A serialized output frame will not use eMIOS_0 channel 21. 1 This bit in the DSPI_A serialized output frame will use eMIOS_0 channel 21.
22 EMIOS0_22_OUT22	eMIOS_0 Channel Select. Select an eMIOS_0 channel to source the serialized DSPI_A output. 0 This bit in the DSPI_A serialized output frame will not use eMIOS_0 channel 22. 1 This bit in the DSPI_A serialized output frame will use eMIOS_0 channel 22.
23 EMIOS0_23_OUT23	eMIOS_0 Channel Select. Select an eMIOS_0 channel to source the serialized DSPI_A output. 0 This bit in the DSPI_A serialized output frame will not use eMIOS_0 channel 23. 1 This bit in the DSPI_A serialized output frame will use eMIOS_0 channel 23.
24 EMIOS0_0_OUT24	eMIOS_0 Channel Select. Select an eMIOS_0 channel to source the serialized DSPI_A output. 0 This bit in the DSPI_A serialized output frame will not use eMIOS_0 channel 0. 1 This bit in the DSPI_A serialized output frame will use eMIOS_0 channel 0.

Table continues on the next page...

SIU_EMIOSA field descriptions (continued)

Field	Description
25 EMIOS0_1_ OUT25	eMIOS_0 Channel Select. Select an eMIOS_0 channel to source the serialized DSPI_A output. 0 This bit in the DSPI_A serialized output frame will not use eMIOS_0 channel 1. 1 This bit in the DSPI_A serialized output frame will use eMIOS_0 channel 1.
26 EMIOS0_2_ OUT26	eMIOS_0 Channel Select. Select an eMIOS_0 channel to source the serialized DSPI_A output. 0 This bit in the DSPI_A serialized output frame will not use eMIOS_0 channel 2. 1 This bit in the DSPI_A serialized output frame will use eMIOS_0 channel 2.
27 EMIOS0_3_ OUT27	eMIOS_0 Channel Select. Select an eMIOS_0 channel to source the serialized DSPI_A output. 0 This bit in the DSPI_A serialized output frame will not use eMIOS_0 channel 3. 1 This bit in the DSPI_A serialized output frame will use eMIOS_0 channel 3.
28 EMIOS0_4_ OUT28	eMIOS_0 Channel Select. Select an eMIOS_0 channel to source the serialized DSPI_A output. 0 This bit in the DSPI_A serialized output frame will not use eMIOS_0 channel 4. 1 This bit in the DSPI_A serialized output frame will use eMIOS_0 channel 4.
29 EMIOS0_5_ OUT29	eMIOS_0 Channel Select. Select an eMIOS_0 channel to source the serialized DSPI_A output. 0 This bit in the DSPI_A serialized output frame will not use eMIOS_0 channel 5. 1 This bit in the DSPI_A serialized output frame will use eMIOS_0 channel 5.
30 EMIOS0_6_ OUT30	eMIOS_0 Channel Select. Select an eMIOS_0 channel to source the serialized DSPI_A output. 0 This bit in the DSPI_A serialized output frame will not use eMIOS_0 channel 6. 1 This bit in the DSPI_A serialized output frame will use eMIOS_0 channel 6.
31 EMIOS0_7_ OUT31	eMIOS_0 Channel Select. Select an eMIOS_0 channel to source the serialized DSPI_A output. 0 This bit in the DSPI_A serialized output frame will not use eMIOS_0 channel 7. 1 This bit in the DSPI_A serialized output frame will use eMIOS_0 channel 7.

8.2.62 Serialized Output Signal Selection Register for DSPI_A (SIU_DSPIAHLA)

Each bit in the DSPI serialized output frame can contain a signal routed from the output of one of three on-chip sources. The sources on this device are the eTPU module, eMIOS module, or a software-updated GPO data register. The mapping between module or data register and the output frame for a single DSPI channel is performed by independent 32-bit registers, one for each module plus one for the data register.

The following register is used by DSPI_A to select the sources of the serialized output when running in DSI or CSI configuration.

Each register bit enables a path from the eTPU_B channel, eMIOS channel and data register bit SIU_DSPIAH/SIU_DSPIAL to the equivalent bit position in the DSPI_A serialized output frame. The user must ensure that bit selections from each of these

Memory Map and Register Definition

registers do not overlap. Multiple sources are logically ORed, which provides the potential for combining outputs from multiple timer channels and data registers to produce more complex bit behavior.

Address: 0h base + D48h offset = D48h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	DSPIAH0	DSPIAH1	DSPIAH2	DSPIAH3	DSPIAH4	DSPIAH5	DSPIAH6	DSPIAH7	DSPIAH8	DSPIAH9	DSPIAH10	DSPIAH11	DSPIAH12	DSPIAH13	DSPIAH14	DSPIAH15
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	DSPIAL16	DSPIAL17	DSPIAL18	DSPIAL19	DSPIAL20	DSPIAL21	DSPIAL22	DSPIAL23	DSPIAL24	DSPIAL25	DSPIAL26	DSPIAL27	DSPIAL28	DSPIAL29	DSPIAL30	DSPIAL31
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

SIU_DSPIAHLA field descriptions

Field	Description
0 DSPIAH0	SIU_DSPIAH Select. Select one bit of the DSPI_A GPO Mask Output High Register to source the serialized DSPI_A output. 0 This bit in the DSPI_A serialized output frame will not use the respective SIU_DSPIAH bit. 1 This bit in the DSPI_A serialized output frame will use the respective SIU_DSPIAH bit.
1 DSPIAH1	SIU_DSPIAH Select. Select one bit of the DSPI_A GPO Mask Output High Register to source the serialized DSPI_A output. 0 This bit in the DSPI_A serialized output frame will not use the respective SIU_DSPIAH bit. 1 This bit in the DSPI_A serialized output frame will use the respective SIU_DSPIAH bit.
2 DSPIAH2	SIU_DSPIAH Select. Select one bit of the DSPI_A GPO Mask Output High Register to source the serialized DSPI_A output. 0 This bit in the DSPI_A serialized output frame will not use the respective SIU_DSPIAH bit. 1 This bit in the DSPI_A serialized output frame will use the respective SIU_DSPIAH bit.
3 DSPIAH3	SIU_DSPIAH Select. Select one bit of the DSPI_A GPO Mask Output High Register to source the serialized DSPI_A output. 0 This bit in the DSPI_A serialized output frame will not use the respective SIU_DSPIAH bit. 1 This bit in the DSPI_A serialized output frame will use the respective SIU_DSPIAH bit.
4 DSPIAH4	SIU_DSPIAH Select. Select one bit of the DSPI_A GPO Mask Output High Register to source the serialized DSPI_A output. 0 This bit in the DSPI_A serialized output frame will not use the respective SIU_DSPIAH bit. 1 This bit in the DSPI_A serialized output frame will use the respective SIU_DSPIAH bit.
5 DSPIAH5	SIU_DSPIAH Select. Select one bit of the DSPI_A GPO Mask Output High Register to source the serialized DSPI_A output.

Table continues on the next page...

SIU_DSPIAHLA field descriptions (continued)

Field	Description
	0 This bit in the DSPI_A serialized output frame will not use the respective SIU_DSPIAH bit. 1 This bit in the DSPI_A serialized output frame will use the respective SIU_DSPIAH bit.
6 DSPIAH6	SIU_DSPIAH Select. Select one bit of the DSPI_A GPO Mask Output High Register to source the serialized DSPI_A output. 0 This bit in the DSPI_A serialized output frame will not use the respective SIU_DSPIAH bit. 1 This bit in the DSPI_A serialized output frame will use the respective SIU_DSPIAH bit.
7 DSPIAH7	SIU_DSPIAH Select. Select one bit of the DSPI_A GPO Mask Output High Register to source the serialized DSPI_A output. 0 This bit in the DSPI_A serialized output frame will not use the respective SIU_DSPIAH bit. 1 This bit in the DSPI_A serialized output frame will use the respective SIU_DSPIAH bit.
8 DSPIAH8	SIU_DSPIAH Select. Select one bit of the DSPI_A GPO Mask Output High Register to source the serialized DSPI_A output. 0 This bit in the DSPI_A serialized output frame will not use the respective SIU_DSPIAH bit. 1 This bit in the DSPI_A serialized output frame will use the respective SIU_DSPIAH bit.
9 DSPIAH9	SIU_DSPIAH Select. Select one bit of the DSPI_A GPO Mask Output High Register to source the serialized DSPI_A output. 0 This bit in the DSPI_A serialized output frame will not use the respective SIU_DSPIAH bit. 1 This bit in the DSPI_A serialized output frame will use the respective SIU_DSPIAH bit.
10 DSPIAH10	SIU_DSPIAH Select. Select one bit of the DSPI_A GPO Mask Output High Register to source the serialized DSPI_A output. 0 This bit in the DSPI_A serialized output frame will not use the respective SIU_DSPIAH bit. 1 This bit in the DSPI_A serialized output frame will use the respective SIU_DSPIAH bit.
11 DSPIAH11	SIU_DSPIAH Select. Select one bit of the DSPI_A GPO Mask Output High Register to source the serialized DSPI_A output. 0 This bit in the DSPI_A serialized output frame will not use the respective SIU_DSPIAH bit. 1 This bit in the DSPI_A serialized output frame will use the respective SIU_DSPIAH bit.
12 DSPIAH12	SIU_DSPIAH Select. Select one bit of the DSPI_A GPO Mask Output High Register to source the serialized DSPI_A output. 0 This bit in the DSPI_A serialized output frame will not use the respective SIU_DSPIAH bit. 1 This bit in the DSPI_A serialized output frame will use the respective SIU_DSPIAH bit.
13 DSPIAH13	SIU_DSPIAH Select. Select one bit of the DSPI_A GPO Mask Output High Register to source the serialized DSPI_A output. 0 This bit in the DSPI_A serialized output frame will not use the respective SIU_DSPIAH bit. 1 This bit in the DSPI_A serialized output frame will use the respective SIU_DSPIAH bit.
14 DSPIAH14	SIU_DSPIAH Select. Select one bit of the DSPI_A GPO Mask Output High Register to source the serialized DSPI_A output. 0 This bit in the DSPI_A serialized output frame will not use the respective SIU_DSPIAH bit. 1 This bit in the DSPI_A serialized output frame will use the respective SIU_DSPIAH bit.
15 DSPIAH15	SIU_DSPIAH Select. Select one bit of the DSPI_A GPO Mask Output High Register to source the serialized DSPI_A output.

Table continues on the next page...

SIU_DSPIAHLA field descriptions (continued)

Field	Description
	0 This bit in the DSPI_A serialized output frame will not use the respective SIU_DSPIAH bit. 1 This bit in the DSPI_A serialized output frame will use the respective SIU_DSPIAH bit.
16 DSPIAL16	SIU_DSPIAL Select. Select one bit of the DSPI_A GPO Mask Output Low Register to source the serialized DSPI_A output. 0 This bit in the DSPI_A serialized output frame will not use the respective SIU_DSPIAL bit. 1 This bit in the DSPI_A serialized output frame will use the respective SIU_DSPIAL bit.
17 DSPIAL17	SIU_DSPIAL Select. Select one bit of the DSPI_A GPO Mask Output Low Register to source the serialized DSPI_A output. 0 This bit in the DSPI_A serialized output frame will not use the respective SIU_DSPIAL bit. 1 This bit in the DSPI_A serialized output frame will use the respective SIU_DSPIAL bit.
18 DSPIAL18	SIU_DSPIAL Select. Select one bit of the DSPI_A GPO Mask Output Low Register to source the serialized DSPI_A output. 0 This bit in the DSPI_A serialized output frame will not use the respective SIU_DSPIAL bit. 1 This bit in the DSPI_A serialized output frame will use the respective SIU_DSPIAL bit.
19 DSPIAL19	SIU_DSPIAL Select. Select one bit of the DSPI_A GPO Mask Output Low Register to source the serialized DSPI_A output. 0 This bit in the DSPI_A serialized output frame will not use the respective SIU_DSPIAL bit. 1 This bit in the DSPI_A serialized output frame will use the respective SIU_DSPIAL bit.
20 DSPIAL20	SIU_DSPIAL Select. Select one bit of the DSPI_A GPO Mask Output Low Register to source the serialized DSPI_A output. 0 This bit in the DSPI_A serialized output frame will not use the respective SIU_DSPIAL bit. 1 This bit in the DSPI_A serialized output frame will use the respective SIU_DSPIAL bit.
21 DSPIAL21	SIU_DSPIAL Select. Select one bit of the DSPI_A GPO Mask Output Low Register to source the serialized DSPI_A output. 0 This bit in the DSPI_A serialized output frame will not use the respective SIU_DSPIAL bit. 1 This bit in the DSPI_A serialized output frame will use the respective SIU_DSPIAL bit.
22 DSPIAL22	SIU_DSPIAL Select. Select one bit of the DSPI_A GPO Mask Output Low Register to source the serialized DSPI_A output. 0 This bit in the DSPI_A serialized output frame will not use the respective SIU_DSPIAL bit. 1 This bit in the DSPI_A serialized output frame will use the respective SIU_DSPIAL bit.
23 DSPIAL23	SIU_DSPIAL Select. Select one bit of the DSPI_A GPO Mask Output Low Register to source the serialized DSPI_A output. 0 This bit in the DSPI_A serialized output frame will not use the respective SIU_DSPIAL bit. 1 This bit in the DSPI_A serialized output frame will use the respective SIU_DSPIAL bit.
24 DSPIAL24	SIU_DSPIAL Select. Select one bit of the DSPI_A GPO Mask Output Low Register to source the serialized DSPI_A output. 0 This bit in the DSPI_A serialized output frame will not use the respective SIU_DSPIAL bit. 1 This bit in the DSPI_A serialized output frame will use the respective SIU_DSPIAL bit.
25 DSPIAL25	SIU_DSPIAL Select. Select one bit of the DSPI_A GPO Mask Output Low Register to source the serialized DSPI_A output.

Table continues on the next page...

SIU_DSPIAHLA field descriptions (continued)

Field	Description
	0 This bit in the DSPI_A serialized output frame will not use the respective SIU_DSPIAL bit. 1 This bit in the DSPI_A serialized output frame will use the respective SIU_DSPIAL bit.
26 DSPIAL26	SIU_DSPIAL Select. Select one bit of the DSPI_A GPO Mask Output Low Register to source the serialized DSPI_A output. 0 This bit in the DSPI_A serialized output frame will not use the respective SIU_DSPIAL bit. 1 This bit in the DSPI_A serialized output frame will use the respective SIU_DSPIAL bit.
27 DSPIAL27	SIU_DSPIAL Select. Select one bit of the DSPI_A GPO Mask Output Low Register to source the serialized DSPI_A output. 0 This bit in the DSPI_A serialized output frame will not use the respective SIU_DSPIAL bit. 1 This bit in the DSPI_A serialized output frame will use the respective SIU_DSPIAL bit.
28 DSPIAL28	SIU_DSPIAL Select. Select one bit of the DSPI_A GPO Mask Output Low Register to source the serialized DSPI_A output. 0 This bit in the DSPI_A serialized output frame will not use the respective SIU_DSPIAL bit. 1 This bit in the DSPI_A serialized output frame will use the respective SIU_DSPIAL bit.
29 DSPIAL29	SIU_DSPIAL Select. Select one bit of the DSPI_A GPO Mask Output Low Register to source the serialized DSPI_A output. 0 This bit in the DSPI_A serialized output frame will not use the respective SIU_DSPIAL bit. 1 This bit in the DSPI_A serialized output frame will use the respective SIU_DSPIAL bit.
30 DSPIAL30	SIU_DSPIAL Select. Select one bit of the DSPI_A GPO Mask Output Low Register to source the serialized DSPI_A output. 0 This bit in the DSPI_A serialized output frame will not use the respective SIU_DSPIAL bit. 1 This bit in the DSPI_A serialized output frame will use the respective SIU_DSPIAL bit.
31 DSPIAL31	SIU_DSPIAL Select. Select one bit of the DSPI_A GPO Mask Output Low Register to source the serialized DSPI_A output. 0 This bit in the DSPI_A serialized output frame will not use the respective SIU_DSPIAL bit. 1 This bit in the DSPI_A serialized output frame will use the respective SIU_DSPIAL bit.

8.2.63 Serialized Output Signal Selection Register for DSPI_B (SIU_ETPUAB)

Each bit in the DSPI serialized output frame can contain a signal routed from the output of one of three on-chip sources. The sources on this device are the eTPU module, eMIOS module, or a software-updated GPO data register. The mapping between module or data register and the output frame for a single DSPI channel is performed by independent 32-bit registers, one for each module plus one for the data register.

The following register is used by DSPI_B to select the sources of the serialized output when running in DSI or CSI configuration.

Memory Map and Register Definition

Each register bit enables a path from the eTPU_A channel, eMIOS channel and data register bit SIU_DSPIBH/SIU_DSPIBL to the equivalent bit position in the DSPI_B serialized output frame. The user must ensure that bit selections from each of these registers do not overlap. Multiple sources are logically ORed, which provides the potential for combining outputs from multiple timer channels and data registers to produce more complex bit behavior.

Address: 0h base + D50h offset = D50h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	ETPUA23	ETPUA22	ETPUA21	ETPUA20	ETPUA19	ETPUA18	ETPUA17	ETPUA16	ETPUA29	ETPUA28	ETPUA27	ETPUA26	ETPUA25	ETPUA24	ETPUA31	ETPUA30
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	ETPUA12	ETPUA13	ETPUA14	ETPUA15	ETPUA0	ETPUA1	ETPUA2	ETPUA3	ETPUA4	ETPUA5	ETPUA6	ETPUA7	ETPUA8	ETPUA9	ETPUA10	ETPUA11
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

SIU_ETPUAB field descriptions

Field	Description
0 ETPUA23	eTPU_A Channel Select. Select an eTPU_A channel to source the serialized DSPI_B output. 0 This bit in the DSPI_B serialized output frame will not use the respective eTPU_A channel. 1 This bit in the DSPI_B serialized output frame will use the respective eTPU_A channel.
1 ETPUA22	eTPU_A Channel Select. Select an eTPU_A channel to source the serialized DSPI_B output. 0 This bit in the DSPI_B serialized output frame will not use the respective eTPU_A channel. 1 This bit in the DSPI_B serialized output frame will use the respective eTPU_A channel.
2 ETPUA21	eTPU_A Channel Select. Select an eTPU_A channel to source the serialized DSPI_B output. 0 This bit in the DSPI_B serialized output frame will not use the respective eTPU_A channel. 1 This bit in the DSPI_B serialized output frame will use the respective eTPU_A channel.
3 ETPUA20	eTPU_A Channel Select. Select an eTPU_A channel to source the serialized DSPI_B output. 0 This bit in the DSPI_B serialized output frame will not use the respective eTPU_A channel. 1 This bit in the DSPI_B serialized output frame will use the respective eTPU_A channel.
4 ETPUA19	eTPU_A Channel Select. Select an eTPU_A channel to source the serialized DSPI_B output. 0 This bit in the DSPI_B serialized output frame will not use the respective eTPU_A channel. 1 This bit in the DSPI_B serialized output frame will use the respective eTPU_A channel.
5 ETPUA18	eTPU_A Channel Select. Select an eTPU_A channel to source the serialized DSPI_B output. 0 This bit in the DSPI_B serialized output frame will not use the respective eTPU_A channel. 1 This bit in the DSPI_B serialized output frame will use the respective eTPU_A channel.

Table continues on the next page...

SIU_ETPUAB field descriptions (continued)

Field	Description
6 ETPUA17	eTPU_A Channel Select. Select an eTPU_A channel to source the serialized DSPI_B output. 0 This bit in the DSPI_B serialized output frame will not use the respective eTPU_A channel. 1 This bit in the DSPI_B serialized output frame will use the respective eTPU_A channel.
7 ETPUA16	eTPU_A Channel Select. Select an eTPU_A channel to source the serialized DSPI_B output. 0 This bit in the DSPI_B serialized output frame will not use the respective eTPU_A channel. 1 This bit in the DSPI_B serialized output frame will use the respective eTPU_A channel.
8 ETPUA29	eTPU_A Channel Select. Select an eTPU_A channel to source the serialized DSPI_B output. 0 This bit in the DSPI_B serialized output frame will not use the respective eTPU_A channel. 1 This bit in the DSPI_B serialized output frame will use the respective eTPU_A channel.
9 ETPUA28	eTPU_A Channel Select. Select an eTPU_A channel to source the serialized DSPI_B output. 0 This bit in the DSPI_B serialized output frame will not use the respective eTPU_A channel. 1 This bit in the DSPI_B serialized output frame will use the respective eTPU_A channel.
10 ETPUA27	eTPU_A Channel Select. Select an eTPU_A channel to source the serialized DSPI_B output. 0 This bit in the DSPI_B serialized output frame will not use the respective eTPU_A channel. 1 This bit in the DSPI_B serialized output frame will use the respective eTPU_A channel.
11 ETPUA26	eTPU_A Channel Select. Select an eTPU_A channel to source the serialized DSPI_B output. 0 This bit in the DSPI_B serialized output frame will not use the respective eTPU_A channel. 1 This bit in the DSPI_B serialized output frame will use the respective eTPU_A channel.
12 ETPUA25	eTPU_A Channel Select. Select an eTPU_A channel to source the serialized DSPI_B output. 0 This bit in the DSPI_B serialized output frame will not use the respective eTPU_A channel. 1 This bit in the DSPI_B serialized output frame will use the respective eTPU_A channel.
13 ETPUA24	eTPU_A Channel Select. Select an eTPU_A channel to source the serialized DSPI_B output. 0 This bit in the DSPI_B serialized output frame will not use the respective eTPU_A channel. 1 This bit in the DSPI_B serialized output frame will use the respective eTPU_A channel.
14 ETPUA31	eTPU_A Channel Select. Select an eTPU_A channel to source the serialized DSPI_B output. 0 This bit in the DSPI_B serialized output frame will not use the respective eTPU_A channel. 1 This bit in the DSPI_B serialized output frame will use the respective eTPU_A channel.
15 ETPUA30	eTPU_A Channel Select. Select an eTPU_A channel to source the serialized DSPI_B output. 0 This bit in the DSPI_B serialized output frame will not use the respective eTPU_A channel. 1 This bit in the DSPI_B serialized output frame will use the respective eTPU_A channel.
16 ETPUA12	eTPU_A Channel Select. Select an eTPU_A channel to source the serialized DSPI_B output. 0 This bit in the DSPI_B serialized output frame will not use the respective eTPU_A channel. 1 This bit in the DSPI_B serialized output frame will use the respective eTPU_A channel.
17 ETPUA13	eTPU_A Channel Select. Select an eTPU_A channel to source the serialized DSPI_B output. 0 This bit in the DSPI_B serialized output frame will not use the respective eTPU_A channel. 1 This bit in the DSPI_B serialized output frame will use the respective eTPU_A channel.

Table continues on the next page...

SIU_ETPUAB field descriptions (continued)

Field	Description
18 ETPUA14	eTPU_A Channel Select. Select an eTPU_A channel to source the serialized DSPI_B output. 0 This bit in the DSPI_B serialized output frame will not use the respective eTPU_A channel. 1 This bit in the DSPI_B serialized output frame will use the respective eTPU_A channel.
19 ETPUA15	eTPU_A Channel Select. Select an eTPU_A channel to source the serialized DSPI_B output. 0 This bit in the DSPI_B serialized output frame will not use the respective eTPU_A channel. 1 This bit in the DSPI_B serialized output frame will use the respective eTPU_A channel.
20 ETPUA0	eTPU_A Channel Select. Select an eTPU_A channel to source the serialized DSPI_B output. 0 This bit in the DSPI_B serialized output frame will not use the respective eTPU_A channel. 1 This bit in the DSPI_B serialized output frame will use the respective eTPU_A channel.
21 ETPUA1	eTPU_A Channel Select. Select an eTPU_A channel to source the serialized DSPI_B output. 0 This bit in the DSPI_B serialized output frame will not use the respective eTPU_A channel. 1 This bit in the DSPI_B serialized output frame will use the respective eTPU_A channel.
22 ETPUA2	eTPU_A Channel Select. Select an eTPU_A channel to source the serialized DSPI_B output. 0 This bit in the DSPI_B serialized output frame will not use the respective eTPU_A channel. 1 This bit in the DSPI_B serialized output frame will use the respective eTPU_A channel.
23 ETPUA3	eTPU_A Channel Select. Select an eTPU_A channel to source the serialized DSPI_B output. 0 This bit in the DSPI_B serialized output frame will not use the respective eTPU_A channel. 1 This bit in the DSPI_B serialized output frame will use the respective eTPU_A channel.
24 ETPUA4	eTPU_A Channel Select. Select an eTPU_A channel to source the serialized DSPI_B output. 0 This bit in the DSPI_B serialized output frame will not use the respective eTPU_A channel. 1 This bit in the DSPI_B serialized output frame will use the respective eTPU_A channel.
25 ETPUA5	eTPU_A Channel Select. Select an eTPU_A channel to source the serialized DSPI_B output. 0 This bit in the DSPI_B serialized output frame will not use the respective eTPU_A channel. 1 This bit in the DSPI_B serialized output frame will use the respective eTPU_A channel.
26 ETPUA6	eTPU_A Channel Select. Select an eTPU_A channel to source the serialized DSPI_B output. 0 This bit in the DSPI_B serialized output frame will not use the respective eTPU_A channel. 1 This bit in the DSPI_B serialized output frame will use the respective eTPU_A channel.
27 ETPUA7	eTPU_A Channel Select. Select an eTPU_A channel to source the serialized DSPI_B output. 0 This bit in the DSPI_B serialized output frame will not use the respective eTPU_A channel. 1 This bit in the DSPI_B serialized output frame will use the respective eTPU_A channel.
28 ETPUA8	eTPU_A Channel Select. Select an eTPU_A channel to source the serialized DSPI_B output. 0 This bit in the DSPI_B serialized output frame will not use the respective eTPU_A channel. 1 This bit in the DSPI_B serialized output frame will use the respective eTPU_A channel.
29 ETPUA9	eTPU_A Channel Select. Select an eTPU_A channel to source the serialized DSPI_B output. 0 This bit in the DSPI_B serialized output frame will not use the respective eTPU_A channel. 1 This bit in the DSPI_B serialized output frame will use the respective eTPU_A channel.

Table continues on the next page...

SIU_ETPUAB field descriptions (continued)

Field	Description
30 ETPUA10	eTPU_A Channel Select. Select an eTPU_A channel to source the serialized DSPI_B output. 0 This bit in the DSPI_B serialized output frame will not use the respective eTPU_A channel. 1 This bit in the DSPI_B serialized output frame will use the respective eTPU_A channel.
31 ETPUA11	eTPU_A Channel Select. Select an eTPU_A channel to source the serialized DSPI_B output. 0 This bit in the DSPI_B serialized output frame will not use the respective eTPU_A channel. 1 This bit in the DSPI_B serialized output frame will use the respective eTPU_A channel.

8.2.64 Serialized Output Signal Selection Register for DSPI_B (SIU_EMIOSB)

Each bit in the DSPI serialized output frame can contain a signal routed from the output of one of three on-chip sources. The sources on this device are the eTPU module, eMIOS module, or a software-updated GPO data register. The mapping between module or data register and the output frame for a single DSPI channel is performed by independent 32-bit registers, one for each module plus one for the data register.

The following register is used by DSPI_B to select the sources of the serialized output when running in DSI or CSI configuration.

Each register bit enables a path from the eTPU_A channel, eMIOS channel and data register bit SIU_DSPIBH/SIU_DSPIBL to the equivalent bit position in the DSPI_B serialized output frame. The user must ensure that bit selections from each of these registers do not overlap. Multiple sources are logically ORed, which provides the potential for combining outputs from multiple timer channels and data registers to produce more complex bit behavior.

Address: 0h base + D54h offset = D54h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	EMIOS1_3_	EMIOS1_2_	EMIOS1_1_	EMIOS1_0_	EMIOS0_6_	EMIOS0_5_	EMIOS0_4_	EMIOS0_3_	EMIOS0_2_	EMIOS0_1_	EMIOS0_0_	EMIOS0_23_	EMIOS1_7_	EMIOS1_6_	EMIOS1_5_	EMIOS1_4_
W	OUT0	OUT1	OUT2	OUT3	OUT4	OUT5	OUT6	OUT7	OUT8	OUT9	OUT10	OUT11	OUT12	OUT13	OUT14	OUT15
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Memory Map and Register Definition

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	EMIOS0_23_	EMIOS1_7_	EMIOS1_6_	EMIOS1_5_	EMIOS1_4_	EMIOS1_3_	EMIOS1_2_	EMIOS1_1_	EMIOS1_0_	EMIOS0_6_	EMIOS0_5_	EMIOS0_4_	EMIOS0_3_	EMIOS0_2_	EMIOS0_1_	EMIOS0_0_
W	OUT16	OUT17	OUT18	OUT19	OUT20	OUT21	OUT22	OUT23	OUT24	OUT25	OUT26	OUT27	OUT28	OUT29	OUT30	OUT31
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

SIU_EMIOSB field descriptions

Field	Description
0 EMIOS1_3_	eMIOS_1 Channel Select. Select an eMIOS_1 channel to source the serialized DSPI_B output. 0 This bit in the DSPI_B serialized output frame will not use eMIOS_1 channel 3. 1 This bit in the DSPI_B serialized output frame will use eMIOS_1 channel 3.
1 EMIOS1_2_	eMIOS_1 Channel Select. Select an eMIOS_1 channel to source the serialized DSPI_B output. 0 This bit in the DSPI_B serialized output frame will not use eMIOS_1 channel 2. 1 This bit in the DSPI_B serialized output frame will use eMIOS_1 channel 2.
2 EMIOS1_1_	eMIOS_1 Channel Select. Select an eMIOS_1 channel to source the serialized DSPI_B output. 0 This bit in the DSPI_B serialized output frame will not use eMIOS_1 channel 1. 1 This bit in the DSPI_B serialized output frame will use eMIOS_1 channel 1.
3 EMIOS1_0_	eMIOS_1 Channel Select. Select an eMIOS_1 channel to source the serialized DSPI_B output. 0 This bit in the DSPI_B serialized output frame will not use eMIOS_1 channel 0. 1 This bit in the DSPI_B serialized output frame will use eMIOS_1 channel 0.
4 EMIOS0_6_	eMIOS_0 Channel Select. Select an eMIOS_0 channel to source the serialized DSPI_B output. 0 This bit in the DSPI_B serialized output frame will not use eMIOS_0 channel 6. 1 This bit in the DSPI_B serialized output frame will use eMIOS_0 channel 6.
5 EMIOS0_5_	eMIOS_0 Channel Select. Select an eMIOS_0 channel to source the serialized DSPI_B output. 0 This bit in the DSPI_B serialized output frame will not use eMIOS_0 channel 5. 1 This bit in the DSPI_B serialized output frame will use eMIOS_0 channel 5.
6 EMIOS0_4_	eMIOS_0 Channel Select. Select an eMIOS_0 channel to source the serialized DSPI_B output. 0 This bit in the DSPI_B serialized output frame will not use eMIOS_0 channel 4. 1 This bit in the DSPI_B serialized output frame will use eMIOS_0 channel 4.
7 EMIOS0_3_	eMIOS_0 Channel Select. Select an eMIOS_0 channel to source the serialized DSPI_B output. 0 This bit in the DSPI_B serialized output frame will not use eMIOS_0 channel 3. 1 This bit in the DSPI_B serialized output frame will use eMIOS_0 channel 3.
8 EMIOS0_2_	eMIOS_0 Channel Select. Select an eMIOS_0 channel to source the serialized DSPI_B output. 0 This bit in the DSPI_B serialized output frame will not use eMIOS_0 channel 2. 1 This bit in the DSPI_B serialized output frame will use eMIOS_0 channel 2.
9 EMIOS0_1_	eMIOS_0 Channel Select. Select an eMIOS_0 channel to source the serialized DSPI_B output. 0 This bit in the DSPI_B serialized output frame will not use eMIOS_0 channel 1. 1 This bit in the DSPI_B serialized output frame will use eMIOS_0 channel 1.

Table continues on the next page...

SIU_EMIOSB field descriptions (continued)

Field	Description
10 EMIOS0_0_ OUT10	eMIOS_0 Channel Select. Select an eMIOS_0 channel to source the serialized DSPI_B output. 0 This bit in the DSPI_B serialized output frame will not use eMIOS_0 channel 0. 1 This bit in the DSPI_B serialized output frame will use eMIOS_0 channel 0.
11 EMIOS0_23_ OUT11	eMIOS_0 Channel Select. Select an eMIOS_0 channel to source the serialized DSPI_B output. 0 This bit in the DSPI_B serialized output frame will not use eMIOS_0 channel 23. 1 This bit in the DSPI_B serialized output frame will use eMIOS_0 channel 23.
12 EMIOS1_7_ OUT12	eMIOS_1 Channel Select. Select an eMIOS_1 channel to source the serialized DSPI_B output. 0 This bit in the DSPI_B serialized output frame will not use eMIOS_1 channel 7. 1 This bit in the DSPI_B serialized output frame will use eMIOS_1 channel 7.
13 EMIOS1_6_ OUT13	eMIOS_1 Channel Select. Select an eMIOS_1 channel to source the serialized DSPI_B output. 0 This bit in the DSPI_B serialized output frame will not use eMIOS_1 channel 6. 1 This bit in the DSPI_B serialized output frame will use eMIOS_1 channel 6.
14 EMIOS1_5_ OUT14	eMIOS_1 Channel Select. Select an eMIOS_1 channel to source the serialized DSPI_B output. 0 This bit in the DSPI_B serialized output frame will not use eMIOS_1 channel 5. 1 This bit in the DSPI_B serialized output frame will use eMIOS_1 channel 5.
15 EMIOS1_4_ OUT15	eMIOS_1 Channel Select. Select an eMIOS_1 channel to source the serialized DSPI_B output. 0 This bit in the DSPI_B serialized output frame will not use eMIOS_1 channel 4. 1 This bit in the DSPI_B serialized output frame will use eMIOS_1 channel 4.
16 EMIOS0_23_ OUT16	eMIOS_0 Channel Select. Select an eMIOS_0 channel to source the serialized DSPI_B output. 0 This bit in the DSPI_B serialized output frame will not use eMIOS_0 channel 23. 1 This bit in the DSPI_B serialized output frame will use eMIOS_0 channel 23.
17 EMIOS1_7_ OUT17	eMIOS_1 Channel Select. Select an eMIOS_1 channel to source the serialized DSPI_B output. 0 This bit in the DSPI_B serialized output frame will not use eMIOS_1 channel 7. 1 This bit in the DSPI_B serialized output frame will use eMIOS_1 channel 7.
18 EMIOS1_6_ OUT18	eMIOS_1 Channel Select. Select an eMIOS_1 channel to source the serialized DSPI_B output. 0 This bit in the DSPI_B serialized output frame will not use eMIOS_1 channel 6. 1 This bit in the DSPI_B serialized output frame will use eMIOS_1 channel 6.
19 EMIOS1_5_ OUT19	eMIOS_1 Channel Select. Select an eMIOS_1 channel to source the serialized DSPI_B output. 0 This bit in the DSPI_B serialized output frame will not use eMIOS_1 channel 5. 1 This bit in the DSPI_B serialized output frame will use eMIOS_1 channel 5.
20 EMIOS1_4_ OUT20	eMIOS_1 Channel Select. Select an eMIOS_1 channel to source the serialized DSPI_B output. 0 This bit in the DSPI_B serialized output frame will not use eMIOS_1 channel 4. 1 This bit in the DSPI_B serialized output frame will use eMIOS_1 channel 4.
21 EMIOS1_3_ OUT21	eMIOS_1 Channel Select. Select an eMIOS_1 channel to source the serialized DSPI_B output. 0 This bit in the DSPI_B serialized output frame will not use eMIOS_1 channel 3. 1 This bit in the DSPI_B serialized output frame will use eMIOS_1 channel 3.

Table continues on the next page...

SIU_EMIOSB field descriptions (continued)

Field	Description
22 EMIOS1_2_ OUT22	eMIOS_1 Channel Select. Select an eMIOS_1 channel to source the serialized DSPI_B output. 0 This bit in the DSPI_B serialized output frame will not use eMIOS_1 channel 2. 1 This bit in the DSPI_B serialized output frame will use eMIOS_1 channel 2.
23 EMIOS1_1_ OUT23	eMIOS_1 Channel Select. Select an eMIOS_1 channel to source the serialized DSPI_B output. 0 This bit in the DSPI_B serialized output frame will not use eMIOS_1 channel 1. 1 This bit in the DSPI_B serialized output frame will use eMIOS_1 channel 1.
24 EMIOS1_0_ OUT24	eMIOS_1 Channel Select. Select an eMIOS_1 channel to source the serialized DSPI_B output. 0 This bit in the DSPI_B serialized output frame will not use eMIOS_1 channel 0. 1 This bit in the DSPI_B serialized output frame will use eMIOS_1 channel 0.
25 EMIOS0_6_ OUT25	eMIOS_0 Channel Select. Select an eMIOS_0 channel to source the serialized DSPI_B output. 0 This bit in the DSPI_B serialized output frame will not use eMIOS_0 channel 6. 1 This bit in the DSPI_B serialized output frame will use eMIOS_0 channel 6.
26 EMIOS0_5_ OUT26	eMIOS_0 Channel Select. Select an eMIOS_0 channel to source the serialized DSPI_B output. 0 This bit in the DSPI_B serialized output frame will not use eMIOS_0 channel 5. 1 This bit in the DSPI_B serialized output frame will use eMIOS_0 channel 5.
27 EMIOS0_4_ OUT27	eMIOS_0 Channel Select. Select an eMIOS_0 channel to source the serialized DSPI_B output. 0 This bit in the DSPI_B serialized output frame will not use eMIOS_0 channel 4. 1 This bit in the DSPI_B serialized output frame will use eMIOS_0 channel 4.
28 EMIOS0_3_ OUT28	eMIOS_0 Channel Select. Select an eMIOS_0 channel to source the serialized DSPI_B output. 0 This bit in the DSPI_B serialized output frame will not use eMIOS_0 channel 3. 1 This bit in the DSPI_B serialized output frame will use eMIOS_0 channel 3.
29 EMIOS0_2_ OUT29	eMIOS_0 Channel Select. Select an eMIOS_0 channel to source the serialized DSPI_B output. 0 This bit in the DSPI_B serialized output frame will not use eMIOS_0 channel 2. 1 This bit in the DSPI_B serialized output frame will use eMIOS_0 channel 2.
30 EMIOS0_1_ OUT30	eMIOS_0 Channel Select. Select an eMIOS_0 channel to source the serialized DSPI_B output. 0 This bit in the DSPI_B serialized output frame will not use eMIOS_0 channel 1. 1 This bit in the DSPI_B serialized output frame will use eMIOS_0 channel 1.
31 EMIOS0_0_ OUT31	eMIOS_0 Channel Select. Select an eMIOS_0 channel to source the serialized DSPI_B output. 0 This bit in the DSPI_B serialized output frame will not use eMIOS_0 channel 0. 1 This bit in the DSPI_B serialized output frame will use eMIOS_0 channel 0.

8.2.65 Serialized Output Signal Selection Register for DSPI_B (SIU_DSPIBHLB)

Each bit in the DSPI serialized output frame can contain a signal routed from the output of one of three on-chip sources. The sources on this device are the eTPU module, eMIOS module, or a software-updated GPO data register. The mapping between module or data register and the output frame for a single DSPI channel is performed by independent 32-bit registers, one for each module plus one for the data register.

The following register is used by DSPI_B to select the sources of the serialized output when running in DSI or CSI configuration.

Each register bit enables a path from the eTPU_A channel, eMIOS channel and data register bit SIU_DSPIBH/SIU_DSPIBL to the equivalent bit position in the DSPI_B serialized output frame. The user must ensure that bit selections from each of these registers do not overlap. Multiple sources are logically ORed, which provides the potential for combining outputs from multiple timer channels and data registers to produce more complex bit behavior.

Address: 0h base + D58h offset = D58h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	DSPIBH0	DSPIBH1	DSPIBH2	DSPIBH3	DSPIBH4	DSPIBH5	DSPIBH6	DSPIBH7	DSPIBH8	DSPIBH9	DSPIBH10	DSPIBH11	DSPIBH12	DSPIBH13	DSPIBH14	DSPIBH15
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	DSPIBL16	DSPIBL17	DSPIBL18	DSPIBL19	DSPIBL20	DSPIBL21	DSPIBL22	DSPIBL23	DSPIBL24	DSPIBL25	DSPIBL26	DSPIBL27	DSPIBL28	DSPIBL29	DSPIBL30	DSPIBL31
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

SIU_DSPIBHLB field descriptions

Field	Description
0 DSPIBH0	SIU_DSPIBH Select. Select one bit of the DSPI_B GPO Mask Output High Register to source the serialized DSPI_B output. 0 This bit in the DSPI_B serialized output frame will not use the respective SIU_DSPIBH bit. 1 This bit in the DSPI_B serialized output frame will use the respective SIU_DSPIBH bit.
1 DSPIBH1	SIU_DSPIBH Select. Select one bit of the DSPI_B GPO Mask Output High Register to source the serialized DSPI_B output.

Table continues on the next page...

SIU_DSPIBHLB field descriptions (continued)

Field	Description
	0 This bit in the DSPI_B serialized output frame will not use the respective SIU_DSPIBH bit. 1 This bit in the DSPI_B serialized output frame will use the respective SIU_DSPIBH bit.
2 DSPIBH2	SIU_DSPIBH Select. Select one bit of the DSPI_B GPO Mask Output High Register to source the serialized DSPI_B output. 0 This bit in the DSPI_B serialized output frame will not use the respective SIU_DSPIBH bit. 1 This bit in the DSPI_B serialized output frame will use the respective SIU_DSPIBH bit.
3 DSPIBH3	SIU_DSPIBH Select. Select one bit of the DSPI_B GPO Mask Output High Register to source the serialized DSPI_B output. 0 This bit in the DSPI_B serialized output frame will not use the respective SIU_DSPIBH bit. 1 This bit in the DSPI_B serialized output frame will use the respective SIU_DSPIBH bit.
4 DSPIBH4	SIU_DSPIBH Select. Select one bit of the DSPI_B GPO Mask Output High Register to source the serialized DSPI_B output. 0 This bit in the DSPI_B serialized output frame will not use the respective SIU_DSPIBH bit. 1 This bit in the DSPI_B serialized output frame will use the respective SIU_DSPIBH bit.
5 DSPIBH5	SIU_DSPIBH Select. Select one bit of the DSPI_B GPO Mask Output High Register to source the serialized DSPI_B output. 0 This bit in the DSPI_B serialized output frame will not use the respective SIU_DSPIBH bit. 1 This bit in the DSPI_B serialized output frame will use the respective SIU_DSPIBH bit.
6 DSPIBH6	SIU_DSPIBH Select. Select one bit of the DSPI_B GPO Mask Output High Register to source the serialized DSPI_B output. 0 This bit in the DSPI_B serialized output frame will not use the respective SIU_DSPIBH bit. 1 This bit in the DSPI_B serialized output frame will use the respective SIU_DSPIBH bit.
7 DSPIBH7	SIU_DSPIBH Select. Select one bit of the DSPI_B GPO Mask Output High Register to source the serialized DSPI_B output. 0 This bit in the DSPI_B serialized output frame will not use the respective SIU_DSPIBH bit. 1 This bit in the DSPI_B serialized output frame will use the respective SIU_DSPIBH bit.
8 DSPIBH8	SIU_DSPIBH Select. Select one bit of the DSPI_B GPO Mask Output High Register to source the serialized DSPI_B output. 0 This bit in the DSPI_B serialized output frame will not use the respective SIU_DSPIBH bit. 1 This bit in the DSPI_B serialized output frame will use the respective SIU_DSPIBH bit.
9 DSPIBH9	SIU_DSPIBH Select. Select one bit of the DSPI_B GPO Mask Output High Register to source the serialized DSPI_B output. 0 This bit in the DSPI_B serialized output frame will not use the respective SIU_DSPIBH bit. 1 This bit in the DSPI_B serialized output frame will use the respective SIU_DSPIBH bit.
10 DSPIBH10	SIU_DSPIBH Select. Select one bit of the DSPI_B GPO Mask Output High Register to source the serialized DSPI_B output. 0 This bit in the DSPI_B serialized output frame will not use the respective SIU_DSPIBH bit. 1 This bit in the DSPI_B serialized output frame will use the respective SIU_DSPIBH bit.
11 DSPIBH11	SIU_DSPIBH Select. Select one bit of the DSPI_B GPO Mask Output High Register to source the serialized DSPI_B output.

Table continues on the next page...

SIU_DSPIBHLB field descriptions (continued)

Field	Description
	0 This bit in the DSPI_B serialized output frame will not use the respective SIU_DSPIBH bit. 1 This bit in the DSPI_B serialized output frame will use the respective SIU_DSPIBH bit.
12 DSPIBH12	SIU_DSPIBH Select. Select one bit of the DSPI_B GPO Mask Output High Register to source the serialized DSPI_B output. 0 This bit in the DSPI_B serialized output frame will not use the respective SIU_DSPIBH bit. 1 This bit in the DSPI_B serialized output frame will use the respective SIU_DSPIBH bit.
13 DSPIBH13	SIU_DSPIBH Select. Select one bit of the DSPI_B GPO Mask Output High Register to source the serialized DSPI_B output. 0 This bit in the DSPI_B serialized output frame will not use the respective SIU_DSPIBH bit. 1 This bit in the DSPI_B serialized output frame will use the respective SIU_DSPIBH bit.
14 DSPIBH14	SIU_DSPIBH Select. Select one bit of the DSPI_B GPO Mask Output High Register to source the serialized DSPI_B output. 0 This bit in the DSPI_B serialized output frame will not use the respective SIU_DSPIBH bit. 1 This bit in the DSPI_B serialized output frame will use the respective SIU_DSPIBH bit.
15 DSPIBH15	SIU_DSPIBH Select. Select one bit of the DSPI_B GPO Mask Output High Register to source the serialized DSPI_B output. 0 This bit in the DSPI_B serialized output frame will not use the respective SIU_DSPIBH bit. 1 This bit in the DSPI_B serialized output frame will use the respective SIU_DSPIBH bit.
16 DSPIBL16	SIU_DSPIBL Select. Select one bit of the DSPI_B GPO Mask Output Low Register to source the serialized DSPI_B output. 0 This bit in the DSPI_B serialized output frame will not use the respective SIU_DSPIBL bit. 1 This bit in the DSPI_B serialized output frame will use the respective SIU_DSPIBL bit.
17 DSPIBL17	SIU_DSPIBL Select. Select one bit of the DSPI_B GPO Mask Output Low Register to source the serialized DSPI_B output. 0 This bit in the DSPI_B serialized output frame will not use the respective SIU_DSPIBL bit. 1 This bit in the DSPI_B serialized output frame will use the respective SIU_DSPIBL bit.
18 DSPIBL18	SIU_DSPIBL Select. Select one bit of the DSPI_B GPO Mask Output Low Register to source the serialized DSPI_B output. 0 This bit in the DSPI_B serialized output frame will not use the respective SIU_DSPIBL bit. 1 This bit in the DSPI_B serialized output frame will use the respective SIU_DSPIBL bit.
19 DSPIBL19	SIU_DSPIBL Select. Select one bit of the DSPI_B GPO Mask Output Low Register to source the serialized DSPI_B output. 0 This bit in the DSPI_B serialized output frame will not use the respective SIU_DSPIBL bit. 1 This bit in the DSPI_B serialized output frame will use the respective SIU_DSPIBL bit.
20 DSPIBL20	SIU_DSPIBL Select. Select one bit of the DSPI_B GPO Mask Output Low Register to source the serialized DSPI_B output. 0 This bit in the DSPI_B serialized output frame will not use the respective SIU_DSPIBL bit. 1 This bit in the DSPI_B serialized output frame will use the respective SIU_DSPIBL bit.
21 DSPIBL21	SIU_DSPIBL Select. Select one bit of the DSPI_B GPO Mask Output Low Register to source the serialized DSPI_B output.

Table continues on the next page...

SIU_DSPIBLB field descriptions (continued)

Field	Description
	0 This bit in the DSPI_B serialized output frame will not use the respective SIU_DSPIBL bit. 1 This bit in the DSPI_B serialized output frame will use the respective SIU_DSPIBL bit.
22 DSPIBL22	SIU_DSPIBL Select. Select one bit of the DSPI_B GPO Mask Output Low Register to source the serialized DSPI_B output. 0 This bit in the DSPI_B serialized output frame will not use the respective SIU_DSPIBL bit. 1 This bit in the DSPI_B serialized output frame will use the respective SIU_DSPIBL bit.
23 DSPIBL23	SIU_DSPIBL Select. Select one bit of the DSPI_B GPO Mask Output Low Register to source the serialized DSPI_B output. 0 This bit in the DSPI_B serialized output frame will not use the respective SIU_DSPIBL bit. 1 This bit in the DSPI_B serialized output frame will use the respective SIU_DSPIBL bit.
24 DSPIBL24	SIU_DSPIBL Select. Select one bit of the DSPI_B GPO Mask Output Low Register to source the serialized DSPI_B output. 0 This bit in the DSPI_B serialized output frame will not use the respective SIU_DSPIBL bit. 1 This bit in the DSPI_B serialized output frame will use the respective SIU_DSPIBL bit.
25 DSPIBL25	SIU_DSPIBL Select. Select one bit of the DSPI_B GPO Mask Output Low Register to source the serialized DSPI_B output. 0 This bit in the DSPI_B serialized output frame will not use the respective SIU_DSPIBL bit. 1 This bit in the DSPI_B serialized output frame will use the respective SIU_DSPIBL bit.
26 DSPIBL26	SIU_DSPIBL Select. Select one bit of the DSPI_B GPO Mask Output Low Register to source the serialized DSPI_B output. 0 This bit in the DSPI_B serialized output frame will not use the respective SIU_DSPIBL bit. 1 This bit in the DSPI_B serialized output frame will use the respective SIU_DSPIBL bit.
27 DSPIBL27	SIU_DSPIBL Select. Select one bit of the DSPI_B GPO Mask Output Low Register to source the serialized DSPI_B output. 0 This bit in the DSPI_B serialized output frame will not use the respective SIU_DSPIBL bit. 1 This bit in the DSPI_B serialized output frame will use the respective SIU_DSPIBL bit.
28 DSPIBL28	SIU_DSPIBL Select. Select one bit of the DSPI_B GPO Mask Output Low Register to source the serialized DSPI_B output. 0 This bit in the DSPI_B serialized output frame will not use the respective SIU_DSPIBL bit. 1 This bit in the DSPI_B serialized output frame will use the respective SIU_DSPIBL bit.
29 DSPIBL29	SIU_DSPIBL Select. Select one bit of the DSPI_B GPO Mask Output Low Register to source the serialized DSPI_B output. 0 This bit in the DSPI_B serialized output frame will not use the respective SIU_DSPIBL bit. 1 This bit in the DSPI_B serialized output frame will use the respective SIU_DSPIBL bit.
30 DSPIBL30	SIU_DSPIBL Select. Select one bit of the DSPI_B GPO Mask Output Low Register to source the serialized DSPI_B output. 0 This bit in the DSPI_B serialized output frame will not use the respective SIU_DSPIBL bit. 1 This bit in the DSPI_B serialized output frame will use the respective SIU_DSPIBL bit.
31 DSPIBL31	SIU_DSPIBL Select. Select one bit of the DSPI_B GPO Mask Output Low Register to source the serialized DSPI_B output.

Table continues on the next page...

SIU_DSPIBHLB field descriptions (continued)

Field	Description
0	This bit in the DSPI_B serialized output frame will not use the respective SIU_DSPIBL bit.
1	This bit in the DSPI_B serialized output frame will use the respective SIU_DSPIBL bit.

8.2.66 Serialized Output Signal Selection Register for DSPI_C (SIU_ETPUAC)

Each bit in the DSPI serialized output frame can contain a signal routed from the output of one of three on-chip sources. The sources on this device are the eTPU module, eMIOS module, or a software-updated GPO data register. The mapping between module or data register and the output frame for a single DSPI channel is performed by independent 32-bit registers, one for each module plus one for the data register.

The following register is used by DSPI_C to select the sources of the serialized output when running in DSI or CSI configuration.

Each register bit enables a path from the eTPU_A/B channel, eMIOS channel and data register bit SIU_DSPICH/SIU_DSPICL to the equivalent bit position in the DSPI_C serialized output frame. The user must ensure that bit selections from each of these registers do not overlap. Multiple sources are logically ORed, which provides the potential for combining outputs from multiple timer channels and data registers to produce more complex bit behavior.

Address: 0h base + D60h offset = D60h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	ETPUA12	ETPUA13	ETPUA14	ETPUA15	ETPUA0	ETPUA1	ETPUA2	ETPUA3	ETPUA4	ETPUA5	ETPUA6	ETPUA7	ETPUA8	ETPUA9	ETPUA10	ETPUA11
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	ETPUA23	ETPUA22	ETPUA21	ETPUA20	ETPUA19	ETPUA18	ETPUA17	ETPUA16	ETPUA29	ETPUA28	ETPUA27	ETPUA26	ETPUA25	ETPUA24	ETPUA31	ETPUA30
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

SIU_ETPUAC field descriptions

Field	Description
0 ETPUA12	eTPU_A Channel Select. Select an eTPU_A channel to source the serialized DSPI_C output.

Table continues on the next page...

SIU_ETPUAC field descriptions (continued)

Field	Description
	0 This bit in the DSPI_C serialized output frame will not use the respective eTPU_A channel. 1 This bit in the DSPI_C serialized output frame will use the respective eTPU_A channel.
1 ETPUA13	eTPU_A Channel Select. Select an eTPU_A channel to source the serialized DSPI_C output. 0 This bit in the DSPI_C serialized output frame will not use the respective eTPU_A channel. 1 This bit in the DSPI_C serialized output frame will use the respective eTPU_A channel.
2 ETPUA14	eTPU_A Channel Select. Select an eTPU_A channel to source the serialized DSPI_C output. 0 This bit in the DSPI_C serialized output frame will not use the respective eTPU_A channel. 1 This bit in the DSPI_C serialized output frame will use the respective eTPU_A channel.
3 ETPUA15	eTPU_A Channel Select. Select an eTPU_A channel to source the serialized DSPI_C output. 0 This bit in the DSPI_C serialized output frame will not use the respective eTPU_A channel. 1 This bit in the DSPI_C serialized output frame will use the respective eTPU_A channel.
4 ETPUA0	eTPU_A Channel Select. Select an eTPU_A channel to source the serialized DSPI_C output. 0 This bit in the DSPI_C serialized output frame will not use the respective eTPU_A channel. 1 This bit in the DSPI_C serialized output frame will use the respective eTPU_A channel.
5 ETPUA1	eTPU_A Channel Select. Select an eTPU_A channel to source the serialized DSPI_C output. 0 This bit in the DSPI_C serialized output frame will not use the respective eTPU_A channel. 1 This bit in the DSPI_C serialized output frame will use the respective eTPU_A channel.
6 ETPUA2	eTPU_A Channel Select. Select an eTPU_A channel to source the serialized DSPI_C output. 0 This bit in the DSPI_C serialized output frame will not use the respective eTPU_A channel. 1 This bit in the DSPI_C serialized output frame will use the respective eTPU_A channel.
7 ETPUA3	eTPU_A Channel Select. Select an eTPU_A channel to source the serialized DSPI_C output. 0 This bit in the DSPI_C serialized output frame will not use the respective eTPU_A channel. 1 This bit in the DSPI_C serialized output frame will use the respective eTPU_A channel.
8 ETPUA4	eTPU_A Channel Select. Select an eTPU_A channel to source the serialized DSPI_C output. 0 This bit in the DSPI_C serialized output frame will not use the respective eTPU_A channel. 1 This bit in the DSPI_C serialized output frame will use the respective eTPU_A channel.
9 ETPUA5	eTPU_A Channel Select. Select an eTPU_A channel to source the serialized DSPI_C output. 0 This bit in the DSPI_C serialized output frame will not use the respective eTPU_A channel. 1 This bit in the DSPI_C serialized output frame will use the respective eTPU_A channel.
10 ETPUA6	eTPU_A Channel Select. Select an eTPU_A channel to source the serialized DSPI_C output. 0 This bit in the DSPI_C serialized output frame will not use the respective eTPU_A channel. 1 This bit in the DSPI_C serialized output frame will use the respective eTPU_A channel.
11 ETPUA7	eTPU_A Channel Select. Select an eTPU_A channel to source the serialized DSPI_C output. 0 This bit in the DSPI_C serialized output frame will not use the respective eTPU_A channel. 1 This bit in the DSPI_C serialized output frame will use the respective eTPU_A channel.
12 ETPUA8	eTPU_A Channel Select. Select an eTPU_A channel to source the serialized DSPI_C output.

Table continues on the next page...

SIU_ETPUAC field descriptions (continued)

Field	Description
	0 This bit in the DSPI_C serialized output frame will not use the respective eTPU_A channel. 1 This bit in the DSPI_C serialized output frame will use the respective eTPU_A channel.
13 ETPUA9	eTPU_A Channel Select. Select an eTPU_A channel to source the serialized DSPI_C output. 0 This bit in the DSPI_C serialized output frame will not use the respective eTPU_A channel. 1 This bit in the DSPI_C serialized output frame will use the respective eTPU_A channel.
14 ETPUA10	eTPU_A Channel Select. Select an eTPU_A channel to source the serialized DSPI_C output. 0 This bit in the DSPI_C serialized output frame will not use the respective eTPU_A channel. 1 This bit in the DSPI_C serialized output frame will use the respective eTPU_A channel.
15 ETPUA11	eTPU_A Channel Select. Select an eTPU_A channel to source the serialized DSPI_C output. 0 This bit in the DSPI_C serialized output frame will not use the respective eTPU_A channel. 1 This bit in the DSPI_C serialized output frame will use the respective eTPU_A channel.
16 ETPUA23	eTPU_A Channel Select. Select an eTPU_A channel to source the serialized DSPI_C output. 0 This bit in the DSPI_C serialized output frame will not use the respective eTPU_A channel. 1 This bit in the DSPI_C serialized output frame will use the respective eTPU_A channel.
17 ETPUA22	eTPU_A Channel Select. Select an eTPU_A channel to source the serialized DSPI_C output. 0 This bit in the DSPI_C serialized output frame will not use the respective eTPU_A channel. 1 This bit in the DSPI_C serialized output frame will use the respective eTPU_A channel.
18 ETPUA21	eTPU_A Channel Select. Select an eTPU_A channel to source the serialized DSPI_C output. 0 This bit in the DSPI_C serialized output frame will not use the respective eTPU_A channel. 1 This bit in the DSPI_C serialized output frame will use the respective eTPU_A channel.
19 ETPUA20	eTPU_A Channel Select. Select an eTPU_A channel to source the serialized DSPI_C output. 0 This bit in the DSPI_C serialized output frame will not use the respective eTPU_A channel. 1 This bit in the DSPI_C serialized output frame will use the respective eTPU_A channel.
20 ETPUA19	eTPU_A Channel Select. Select an eTPU_A channel to source the serialized DSPI_C output. 0 This bit in the DSPI_C serialized output frame will not use the respective eTPU_A channel. 1 This bit in the DSPI_C serialized output frame will use the respective eTPU_A channel.
21 ETPUA18	eTPU_A Channel Select. Select an eTPU_A channel to source the serialized DSPI_C output. 0 This bit in the DSPI_C serialized output frame will not use the respective eTPU_A channel. 1 This bit in the DSPI_C serialized output frame will use the respective eTPU_A channel.
22 ETPUA17	eTPU_A Channel Select. Select an eTPU_A channel to source the serialized DSPI_C output. 0 This bit in the DSPI_C serialized output frame will not use the respective eTPU_A channel. 1 This bit in the DSPI_C serialized output frame will use the respective eTPU_A channel.
23 ETPUA16	eTPU_A Channel Select. Select an eTPU_A channel to source the serialized DSPI_C output. 0 This bit in the DSPI_C serialized output frame will not use the respective eTPU_A channel. 1 This bit in the DSPI_C serialized output frame will use the respective eTPU_A channel.
24 ETPUA29	eTPU_A Channel Select. Select an eTPU_A channel to source the serialized DSPI_C output.

Table continues on the next page...

SIU_ETPUAC field descriptions (continued)

Field	Description
	0 This bit in the DSPI_C serialized output frame will not use the respective eTPU_A channel. 1 This bit in the DSPI_C serialized output frame will use the respective eTPU_A channel.
25 ETPUA28	eTPU_A Channel Select. Select an eTPU_A channel to source the serialized DSPI_C output. 0 This bit in the DSPI_C serialized output frame will not use the respective eTPU_A channel. 1 This bit in the DSPI_C serialized output frame will use the respective eTPU_A channel.
26 ETPUA27	eTPU_A Channel Select. Select an eTPU_A channel to source the serialized DSPI_C output. 0 This bit in the DSPI_C serialized output frame will not use the respective eTPU_A channel. 1 This bit in the DSPI_C serialized output frame will use the respective eTPU_A channel.
27 ETPUA26	eTPU_A Channel Select. Select an eTPU_A channel to source the serialized DSPI_C output. 0 This bit in the DSPI_C serialized output frame will not use the respective eTPU_A channel. 1 This bit in the DSPI_C serialized output frame will use the respective eTPU_A channel.
28 ETPUA25	eTPU_A Channel Select. Select an eTPU_A channel to source the serialized DSPI_C output. 0 This bit in the DSPI_C serialized output frame will not use the respective eTPU_A channel. 1 This bit in the DSPI_C serialized output frame will use the respective eTPU_A channel.
29 ETPUA24	eTPU_A Channel Select. Select an eTPU_A channel to source the serialized DSPI_C output. 0 This bit in the DSPI_C serialized output frame will not use the respective eTPU_A channel. 1 This bit in the DSPI_C serialized output frame will use the respective eTPU_A channel.
30 ETPUA31	eTPU_A Channel Select. Select an eTPU_A channel to source the serialized DSPI_C output. 0 This bit in the DSPI_C serialized output frame will not use the respective eTPU_A channel. 1 This bit in the DSPI_C serialized output frame will use the respective eTPU_A channel.
31 ETPUA30	eTPU_A Channel Select. Select an eTPU_A channel to source the serialized DSPI_C output. 0 This bit in the DSPI_C serialized output frame will not use the respective eTPU_A channel. 1 This bit in the DSPI_C serialized output frame will use the respective eTPU_A channel.

8.2.67 Serialized Output Signal Selection Register for DSPI_C (SIU_EMIOSC)

Each bit in the DSPI serialized output frame can contain a signal routed from the output of one of three on-chip sources. The sources on this device are the eTPU module, eMIOS module, or a software-updated GPO data register. The mapping between module or data register and the output frame for a single DSPI channel is performed by independent 32-bit registers, one for each module plus one for the data register.

The following register is used by DSPI_C to select the sources of the serialized output when running in DSI or CSI configuration.

Each register bit enables a path from the eTPU_A/B channel, eMIOS channel and data register bit SIU_DSPICH/SIU_DSPICL to the equivalent bit position in the DSPI_C serialized output frame. The user must ensure that bit selections from each of these registers do not overlap. Multiple sources are logically ORed, which provides the potential for combining outputs from multiple timer channels and data registers to produce more complex bit behavior.

Address: 0h base + D64h offset = D64h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	EMIOS1_4_	EMIOS1_5_	EMIOS1_6_	EMIOS1_7_	EMIOS0_23_	EMIOS0_0_	EMIOS0_1_	EMIOS0_2_	EMIOS0_3_	EMIOS0_4_	EMIOS0_5_	EMIOS0_6_	EMIOS1_0_	EMIOS1_1_	EMIOS1_2_	EMIOS1_3_
W	OUT0	OUT1	OUT2	OUT3	OUT4	OUT5	OUT6	OUT7	OUT8	OUT9	OUT10	OUT11	OUT12	OUT13	OUT14	OUT15
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	EMIOS0_0_	EMIOS0_1_	EMIOS0_2_	EMIOS0_3_	EMIOS0_4_	EMIOS0_5_	EMIOS0_6_	EMIOS1_0_	EMIOS1_1_	EMIOS1_2_	EMIOS1_3_	EMIOS1_4_	EMIOS1_5_	EMIOS1_6_	EMIOS1_7_	EMIOS0_23_
W	OUT16	OUT17	OUT18	OUT19	OUT20	OUT21	OUT22	OUT23	OUT24	OUT25	OUT26	OUT27	OUT28	OUT29	OUT30	OUT31
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

SIU_EMIOSC field descriptions

Field	Description
0 EMIOS1_4_ OUT0	eMIOS_1 Channel Select. Select an eMIOS_1 channel to source the serialized DSPI_C output. 0 This bit in the DSPI_C serialized output frame will not use eMIOS_1 channel 4. 1 This bit in the DSPI_C serialized output frame will use eMIOS_1 channel 4.
1 EMIOS1_5_ OUT1	eMIOS_1 Channel Select. Select an eMIOS_1 channel to source the serialized DSPI_C output. 0 This bit in the DSPI_C serialized output frame will not use eMIOS_1 channel 5. 1 This bit in the DSPI_C serialized output frame will use eMIOS_1 channel .5
2 EMIOS1_6_ OUT2	eMIOS_1 Channel Select. Select an eMIOS_1 channel to source the serialized DSPI_C output. 0 This bit in the DSPI_C serialized output frame will not use eMIOS_1 channel 6. 1 This bit in the DSPI_C serialized output frame will use eMIOS_1 channel 6.
3 EMIOS1_7_ OUT3	eMIOS_1 Channel Select. Select an eMIOS_1 channel to source the serialized DSPI_C output. 0 This bit in the DSPI_C serialized output frame will not use eMIOS_1 channel 7. 1 This bit in the DSPI_C serialized output frame will use eMIOS_1 channel 7.
4 EMIOS0_23_ OUT4	eMIOS_0 Channel Select. Select an eMIOS_0 channel to source the serialized DSPI_C output. 0 This bit in the DSPI_C serialized output frame will not use eMIOS_0 channel 23. 1 This bit in the DSPI_C serialized output frame will use eMIOS_0 channel 23.

Table continues on the next page...

SIU_EMIOSC field descriptions (continued)

Field	Description
5 EMIOS0_0_OUT5	eMIOS_0 Channel Select. Select an eMIOS_0 channel to source the serialized DSPI_C output. 0 This bit in the DSPI_C serialized output frame will not use eMIOS_0 channel 0. 1 This bit in the DSPI_C serialized output frame will use eMIOS_0 channel 0.
6 EMIOS0_1_OUT6	eMIOS_0 Channel Select. Select an eMIOS_0 channel to source the serialized DSPI_C output. 0 This bit in the DSPI_C serialized output frame will not use eMIOS_0 channel 1. 1 This bit in the DSPI_C serialized output frame will use eMIOS_0 channel 1.
7 EMIOS0_2_OUT7	eMIOS_0 Channel Select. Select an eMIOS_0 channel to source the serialized DSPI_C output. 0 This bit in the DSPI_C serialized output frame will not use eMIOS_0 channel 2. 1 This bit in the DSPI_C serialized output frame will use eMIOS_0 channel 2.
8 EMIOS0_3_OUT8	eMIOS_0 Channel Select. Select an eMIOS_0 channel to source the serialized DSPI_C output. 0 This bit in the DSPI_C serialized output frame will not use eMIOS_0 channel 3. 1 This bit in the DSPI_C serialized output frame will use eMIOS_0 channel 3.
9 EMIOS0_4_OUT9	eMIOS_0 Channel Select. Select an eMIOS_0 channel to source the serialized DSPI_C output. 0 This bit in the DSPI_C serialized output frame will not use eMIOS_0 channel 4. 1 This bit in the DSPI_C serialized output frame will use eMIOS_0 channel 4.
10 EMIOS0_5_OUT10	eMIOS_0 Channel Select. Select an eMIOS_0 channel to source the serialized DSPI_C output. 0 This bit in the DSPI_C serialized output frame will not use eMIOS_0 channel 5. 1 This bit in the DSPI_C serialized output frame will use eMIOS_0 channel 5.
11 EMIOS0_6_OUT11	eMIOS_0 Channel Select. Select an eMIOS_0 channel to source the serialized DSPI_C output. 0 This bit in the DSPI_C serialized output frame will not use eMIOS_0 channel 6. 1 This bit in the DSPI_C serialized output frame will use eMIOS_0 channel 6.
12 EMIOS1_0_OUT12	eMIOS_1 Channel Select. Select an eMIOS_1 channel to source the serialized DSPI_C output. 0 This bit in the DSPI_C serialized output frame will not use eMIOS_1 channel 0. 1 This bit in the DSPI_C serialized output frame will use eMIOS_1 channel 0.
13 EMIOS1_1_OUT13	eMIOS_1 Channel Select. Select an eMIOS_1 channel to source the serialized DSPI_C output. 0 This bit in the DSPI_C serialized output frame will not use eMIOS_1 channel 1. 1 This bit in the DSPI_C serialized output frame will use eMIOS_1 channel 1.
14 EMIOS1_2_OUT14	eMIOS_1 Channel Select. Select an eMIOS_1 channel to source the serialized DSPI_C output. 0 This bit in the DSPI_C serialized output frame will not use eMIOS_1 channel 2. 1 This bit in the DSPI_C serialized output frame will use eMIOS_1 channel 2.
15 EMIOS1_3_OUT15	eMIOS_1 Channel Select. Select an eMIOS_1 channel to source the serialized DSPI_C output. 0 This bit in the DSPI_C serialized output frame will not use eMIOS_1 channel 3. 1 This bit in the DSPI_C serialized output frame will use eMIOS_1 channel 3.
16 EMIOS0_0_OUT16	eMIOS_0 Channel Select. Select an eMIOS_0 channel to source the serialized DSPI_C output. 0 This bit in the DSPI_C serialized output frame will not use eMIOS_0 channel 0. 1 This bit in the DSPI_C serialized output frame will use eMIOS_0 channel 0.

Table continues on the next page...

SIU_EMIOSC field descriptions (continued)

Field	Description
17 EMIOS0_1_ OUT17	eMIOS_0 Channel Select. Select an eMIOS_0 channel to source the serialized DSPI_C output. 0 This bit in the DSPI_C serialized output frame will not use eMIOS_0 channel 1. 1 This bit in the DSPI_C serialized output frame will use eMIOS_0 channel 1.
18 EMIOS0_2_ OUT18	eMIOS_0 Channel Select. Select an eMIOS_0 channel to source the serialized DSPI_C output. 0 This bit in the DSPI_C serialized output frame will not use eMIOS_0 channel 2. 1 This bit in the DSPI_C serialized output frame will use eMIOS_0 channel 2.
19 EMIOS0_3_ OUT19	eMIOS_0 Channel Select. Select an eMIOS_0 channel to source the serialized DSPI_C output. 0 This bit in the DSPI_C serialized output frame will not use eMIOS_0 channel 3. 1 This bit in the DSPI_C serialized output frame will use eMIOS_0 channel 3.
20 EMIOS0_4_ OUT20	eMIOS_0 Channel Select. Select an eMIOS_0 channel to source the serialized DSPI_C output. 0 This bit in the DSPI_C serialized output frame will not use eMIOS_0 channel 4. 1 This bit in the DSPI_C serialized output frame will use eMIOS_0 channel 4.
21 EMIOS0_5_ OUT21	eMIOS_0 Channel Select. Select an eMIOS_0 channel to source the serialized DSPI_C output. 0 This bit in the DSPI_C serialized output frame will not use eMIOS_0 channel 5. 1 This bit in the DSPI_C serialized output frame will use eMIOS_0 channel 5.
22 EMIOS0_6_ OUT22	eMIOS_0 Channel Select. Select an eMIOS_0 channel to source the serialized DSPI_C output. 0 This bit in the DSPI_C serialized output frame will not use eMIOS_0 channel 6. 1 This bit in the DSPI_C serialized output frame will use eMIOS_0 channel 6.
23 EMIOS1_0_ OUT23	eMIOS_1 Channel Select. Select an eMIOS_1 channel to source the serialized DSPI_C output. 0 This bit in the DSPI_C serialized output frame will not use eMIOS_1 channel 0. 1 This bit in the DSPI_C serialized output frame will use eMIOS_1 channel 0.
24 EMIOS1_1_ OUT24	eMIOS_1 Channel Select. Select an eMIOS_1 channel to source the serialized DSPI_C output. 0 This bit in the DSPI_C serialized output frame will not use eMIOS_1 channel 1. 1 This bit in the DSPI_C serialized output frame will use eMIOS_1 channel 1.
25 EMIOS1_2_ OUT25	eMIOS_1 Channel Select. Select an eMIOS_1 channel to source the serialized DSPI_C output. 0 This bit in the DSPI_C serialized output frame will not use eMIOS_1 channel 2. 1 This bit in the DSPI_C serialized output frame will use eMIOS_1 channel 2.
26 EMIOS1_3_ OUT26	eMIOS_1 Channel Select. Select an eMIOS_1 channel to source the serialized DSPI_C output. 0 This bit in the DSPI_C serialized output frame will not use eMIOS_1 channel 3. 1 This bit in the DSPI_C serialized output frame will use eMIOS_1 channel 3.
27 EMIOS1_4_ OUT27	eMIOS_1 Channel Select. Select an eMIOS_1 channel to source the serialized DSPI_C output. 0 This bit in the DSPI_C serialized output frame will not use eMIOS_1 channel 4. 1 This bit in the DSPI_C serialized output frame will use eMIOS_1 channel 4.
28 EMIOS1_5_ OUT28	eMIOS_1 Channel Select. Select an eMIOS_1 channel to source the serialized DSPI_C output. 0 This bit in the DSPI_C serialized output frame will not use eMIOS_1 channel 5. 1 This bit in the DSPI_C serialized output frame will use eMIOS_1 channel 5.

Table continues on the next page...

SIU_EMIOSC field descriptions (continued)

Field	Description
29 EMIOS1_6_ OUT29	eMIOS_1 Channel Select. Select an eMIOS_1 channel to source the serialized DSPI_C output. 0 This bit in the DSPI_C serialized output frame will not use eMIOS_1 channel 6. 1 This bit in the DSPI_C serialized output frame will use eMIOS_1 channel 6.
30 EMIOS1_7_ OUT30	eMIOS_1 Channel Select. Select an eMIOS_1 channel to source the serialized DSPI_C output. 0 This bit in the DSPI_C serialized output frame will not use eMIOS_1 channel 7. 1 This bit in the DSPI_C serialized output frame will use eMIOS_1 channel 7.
31 EMIOS0_23_ OUT31	eMIOS_0 Channel Select. Select an eMIOS_0 channel to source the serialized DSPI_C output. 0 This bit in the DSPI_C serialized output frame will not use eMIOS_0 channel 23. 1 This bit in the DSPI_C serialized output frame will use eMIOS_0 channel 23.

8.2.68 Serialized Output Signal Selection Register for DSPI_C (SIU_DSPICHL)

Each bit in the DSPI serialized output frame can contain a signal routed from the output of one of three on-chip sources. The sources on this device are the eTPU module, eMIOS module, or a software-updated GPO data register. The mapping between module or data register and the output frame for a single DSPI channel is performed by independent 32-bit registers, one for each module plus one for the data register.

The following register is used by DSPI_C to select the sources of the serialized output when running in DSI or CSI configuration.

Each register bit enables a path from the eTPU_A/B channel, eMIOS channel and data register bit SIU_DSPICH/SIU_DSPICL to the equivalent bit position in the DSPI_C serialized output frame. The user must ensure that bit selections from each of these registers do not overlap. Multiple sources are logically ORed, which provides the potential for combining outputs from multiple timer channels and data registers to produce more complex bit behavior.

Address: 0h base + D68h offset = D68h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	DSPICH0	DSPICH1	DSPICH2	DSPICH3	DSPICH4	DSPICH5	DSPICH6	DSPICH7	DSPICH8	DSPICH9	DSPICH10	DSPICH11	DSPICH12	DSPICH13	DSPICH14	DSPICH15
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	DSPICL16	DSPICL17	DSPICL18	DSPICL19	DSPICL20	DSPICL21	DSPICL22	DSPICL23	DSPICL24	DSPICL25	DSPICL26	DSPICL27	DSPICL28	DSPICL29	DSPICL30	DSPICL31
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

SIU_DSPICHL field descriptions

Field	Description
0 DSPICHL0	SIU_DSPICHL Select. Select one bit of the DSPI_C GPO Mask Output High Register to source the serialized DSPI_C output. 0 This bit in the DSPI_C serialized output frame will not use the respective SIU_DSPICHL bit. 1 This bit in the DSPI_C serialized output frame will use the respective SIU_DSPICHL bit.
1 DSPICHL1	SIU_DSPICHL Select. Select one bit of the DSPI_C GPO Mask Output High Register to source the serialized DSPI_C output. 0 This bit in the DSPI_C serialized output frame will not use the respective SIU_DSPICHL bit. 1 This bit in the DSPI_C serialized output frame will use the respective SIU_DSPICHL bit.
2 DSPICHL2	SIU_DSPICHL Select. Select one bit of the DSPI_C GPO Mask Output High Register to source the serialized DSPI_C output. 0 This bit in the DSPI_C serialized output frame will not use the respective SIU_DSPICHL bit. 1 This bit in the DSPI_C serialized output frame will use the respective SIU_DSPICHL bit.
3 DSPICHL3	SIU_DSPICHL Select. Select one bit of the DSPI_C GPO Mask Output High Register to source the serialized DSPI_C output. 0 This bit in the DSPI_C serialized output frame will not use the respective SIU_DSPICHL bit. 1 This bit in the DSPI_C serialized output frame will use the respective SIU_DSPICHL bit.
4 DSPICHL4	SIU_DSPICHL Select. Select one bit of the DSPI_C GPO Mask Output High Register to source the serialized DSPI_C output. 0 This bit in the DSPI_C serialized output frame will not use the respective SIU_DSPICHL bit. 1 This bit in the DSPI_C serialized output frame will use the respective SIU_DSPICHL bit.
5 DSPICHL5	SIU_DSPICHL Select. Select one bit of the DSPI_C GPO Mask Output High Register to source the serialized DSPI_C output. 0 This bit in the DSPI_C serialized output frame will not use the respective SIU_DSPICHL bit. 1 This bit in the DSPI_C serialized output frame will use the respective SIU_DSPICHL bit.
6 DSPICHL6	SIU_DSPICHL Select. Select one bit of the DSPI_C GPO Mask Output High Register to source the serialized DSPI_C output. 0 This bit in the DSPI_C serialized output frame will not use the respective SIU_DSPICHL bit. 1 This bit in the DSPI_C serialized output frame will use the respective SIU_DSPICHL bit.
7 DSPICHL7	SIU_DSPICHL Select. Select one bit of the DSPI_C GPO Mask Output High Register to source the serialized DSPI_C output. 0 This bit in the DSPI_C serialized output frame will not use the respective SIU_DSPICHL bit. 1 This bit in the DSPI_C serialized output frame will use the respective SIU_DSPICHL bit.
8 DSPICHL8	SIU_DSPICHL Select. Select one bit of the DSPI_C GPO Mask Output High Register to source the serialized DSPI_C output.

Table continues on the next page...

SIU_DSPICHL C field descriptions (continued)

Field	Description
	0 This bit in the DSPI_C serialized output frame will not use the respective SIU_DSPICH bit. 1 This bit in the DSPI_C serialized output frame will use the respective SIU_DSPICH bit.
9 DSPICH9	SIU_DSPICH Select. Select one bit of the DSPI_C GPO Mask Output High Register to source the serialized DSPI_C output. 0 This bit in the DSPI_C serialized output frame will not use the respective SIU_DSPICH bit. 1 This bit in the DSPI_C serialized output frame will use the respective SIU_DSPICH bit.
10 DSPICH10	SIU_DSPICH Select. Select one bit of the DSPI_C GPO Mask Output High Register to source the serialized DSPI_C output. 0 This bit in the DSPI_C serialized output frame will not use the respective SIU_DSPICH bit. 1 This bit in the DSPI_C serialized output frame will use the respective SIU_DSPICH bit.
11 DSPICH11	SIU_DSPICH Select. Select one bit of the DSPI_C GPO Mask Output High Register to source the serialized DSPI_C output. 0 This bit in the DSPI_C serialized output frame will not use the respective SIU_DSPICH bit. 1 This bit in the DSPI_C serialized output frame will use the respective SIU_DSPICH bit.
12 DSPICH12	SIU_DSPICH Select. Select one bit of the DSPI_C GPO Mask Output High Register to source the serialized DSPI_C output. 0 This bit in the DSPI_C serialized output frame will not use the respective SIU_DSPICH bit. 1 This bit in the DSPI_C serialized output frame will use the respective SIU_DSPICH bit.
13 DSPICH13	SIU_DSPICH Select. Select one bit of the DSPI_C GPO Mask Output High Register to source the serialized DSPI_C output. 0 This bit in the DSPI_C serialized output frame will not use the respective SIU_DSPICH bit. 1 This bit in the DSPI_C serialized output frame will use the respective SIU_DSPICH bit.
14 DSPICH14	SIU_DSPICH Select. Select one bit of the DSPI_C GPO Mask Output High Register to source the serialized DSPI_C output. 0 This bit in the DSPI_C serialized output frame will not use the respective SIU_DSPICH bit. 1 This bit in the DSPI_C serialized output frame will use the respective SIU_DSPICH bit.
15 DSPICH15	SIU_DSPICH Select. Select one bit of the DSPI_C GPO Mask Output High Register to source the serialized DSPI_C output. 0 This bit in the DSPI_C serialized output frame will not use the respective SIU_DSPICH bit. 1 This bit in the DSPI_C serialized output frame will use the respective SIU_DSPICH bit.
16 DSPICL16	SIU_DSPICL Select. Select one bit of the DSPI_C GPO Mask Output Low Register to source the serialized DSPI_C output. 0 This bit in the DSPI_C serialized output frame will not use the respective SIU_DSPICL bit. 1 This bit in the DSPI_C serialized output frame will use the respective SIU_DSPICL bit.
17 DSPICL17	SIU_DSPICL Select. Select one bit of the DSPI_C GPO Mask Output Low Register to source the serialized DSPI_C output. 0 This bit in the DSPI_C serialized output frame will not use the respective SIU_DSPICL bit. 1 This bit in the DSPI_C serialized output frame will use the respective SIU_DSPICL bit.
18 DSPICL18	SIU_DSPICL Select. Select one bit of the DSPI_C GPO Mask Output Low Register to source the serialized DSPI_C output.

Table continues on the next page...

SIU_DSPICHL field descriptions (continued)

Field	Description
	0 This bit in the DSPI_C serialized output frame will not use the respective SIU_DSPICL bit. 1 This bit in the DSPI_C serialized output frame will use the respective SIU_DSPICL bit.
19 DSPICL19	SIU_DSPICL Select. Select one bit of the DSPI_C GPO Mask Output Low Register to source the serialized DSPI_C output. 0 This bit in the DSPI_C serialized output frame will not use the respective SIU_DSPICL bit. 1 This bit in the DSPI_C serialized output frame will use the respective SIU_DSPICL bit.
20 DSPICL20	SIU_DSPICL Select. Select one bit of the DSPI_C GPO Mask Output Low Register to source the serialized DSPI_C output. 0 This bit in the DSPI_C serialized output frame will not use the respective SIU_DSPICL bit. 1 This bit in the DSPI_C serialized output frame will use the respective SIU_DSPICL bit.
21 DSPICL21	SIU_DSPICL Select. Select one bit of the DSPI_C GPO Mask Output Low Register to source the serialized DSPI_C output. 0 This bit in the DSPI_C serialized output frame will not use the respective SIU_DSPICL bit. 1 This bit in the DSPI_C serialized output frame will use the respective SIU_DSPICL bit.
22 DSPICL22	SIU_DSPICL Select. Select one bit of the DSPI_C GPO Mask Output Low Register to source the serialized DSPI_C output. 0 This bit in the DSPI_C serialized output frame will not use the respective SIU_DSPICL bit. 1 This bit in the DSPI_C serialized output frame will use the respective SIU_DSPICL bit.
23 DSPICL23	SIU_DSPICL Select. Select one bit of the DSPI_C GPO Mask Output Low Register to source the serialized DSPI_C output. 0 This bit in the DSPI_C serialized output frame will not use the respective SIU_DSPICL bit. 1 This bit in the DSPI_C serialized output frame will use the respective SIU_DSPICL bit.
24 DSPICL24	SIU_DSPICL Select. Select one bit of the DSPI_C GPO Mask Output Low Register to source the serialized DSPI_C output. 0 This bit in the DSPI_C serialized output frame will not use the respective SIU_DSPICL bit. 1 This bit in the DSPI_C serialized output frame will use the respective SIU_DSPICL bit.
25 DSPICL25	SIU_DSPICL Select. Select one bit of the DSPI_C GPO Mask Output Low Register to source the serialized DSPI_C output. 0 This bit in the DSPI_C serialized output frame will not use the respective SIU_DSPICL bit. 1 This bit in the DSPI_C serialized output frame will use the respective SIU_DSPICL bit.
26 DSPICL26	SIU_DSPICL Select. Select one bit of the DSPI_C GPO Mask Output Low Register to source the serialized DSPI_C output. 0 This bit in the DSPI_C serialized output frame will not use the respective SIU_DSPICL bit. 1 This bit in the DSPI_C serialized output frame will use the respective SIU_DSPICL bit.
27 DSPICL27	SIU_DSPICL Select. Select one bit of the DSPI_C GPO Mask Output Low Register to source the serialized DSPI_C output. 0 This bit in the DSPI_C serialized output frame will not use the respective SIU_DSPICL bit. 1 This bit in the DSPI_C serialized output frame will use the respective SIU_DSPICL bit.
28 DSPICL28	SIU_DSPICL Select. Select one bit of the DSPI_C GPO Mask Output Low Register to source the serialized DSPI_C output.

Table continues on the next page...

SIU_DSPICHL field descriptions (continued)

Field	Description
	0 This bit in the DSPI_C serialized output frame will not use the respective SIU_DSPICL bit. 1 This bit in the DSPI_C serialized output frame will use the respective SIU_DSPICL bit.
29 DSPICL29	SIU_DSPICL Select. Select one bit of the DSPI_C GPO Mask Output Low Register to source the serialized DSPI_C output. 0 This bit in the DSPI_C serialized output frame will not use the respective SIU_DSPICL bit. 1 This bit in the DSPI_C serialized output frame will use the respective SIU_DSPICL bit.
30 DSPICL30	SIU_DSPICL Select. Select one bit of the DSPI_C GPO Mask Output Low Register to source the serialized DSPI_C output. 0 This bit in the DSPI_C serialized output frame will not use the respective SIU_DSPICL bit. 1 This bit in the DSPI_C serialized output frame will use the respective SIU_DSPICL bit.
31 DSPICL31	SIU_DSPICL Select. Select one bit of the DSPI_C GPO Mask Output Low Register to source the serialized DSPI_C output. 0 This bit in the DSPI_C serialized output frame will not use the respective SIU_DSPICL bit. 1 This bit in the DSPI_C serialized output frame will use the respective SIU_DSPICL bit.

8.2.69 Serialized Output Signal Selection Register for DSPI_C (SIU_ETPUBC)

Each bit in the DSPI serialized output frame can contain a signal routed from the output of one of three on-chip sources. The sources on this device are the eTPU module, eMIOS module, or a software-updated GPO data register. The mapping between module or data register and the output frame for a single DSPI channel is performed by independent 32-bit registers, one for each module plus one for the data register.

The following register is used by DSPI_C to select the sources of the serialized output when running in DSI or CSI configuration.

Each register bit enables a path from the eTPU_A/B channel, eMIOS channel and data register bit SIU_DSPICH/SIU_DSPICL to the equivalent bit position in the DSPI_C serialized output frame. The user must ensure that bit selections from each of these registers do not overlap. Multiple sources are logically ORed, which provides the potential for combining outputs from multiple timer channels and data registers to produce more complex bit behavior.

Address: 0h base + D6Ch offset = D6Ch

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	ETPUB15	ETPUB14	ETPUB13	ETPUB12	ETPUB11	ETPUB10	ETPUB9	ETPUB8	ETPUB7	ETPUB6	ETPUB5	ETPUB4	ETPUB3	ETPUB2	ETPUB1	ETPUB0
W	ETPUB15	ETPUB14	ETPUB13	ETPUB12	ETPUB11	ETPUB10	ETPUB9	ETPUB8	ETPUB7	ETPUB6	ETPUB5	ETPUB4	ETPUB3	ETPUB2	ETPUB1	ETPUB0
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	ETPUB31	ETPUB30	ETPUB29	ETPUB28	ETPUB27	ETPUB26	ETPUB25	ETPUB24	ETPUB23	ETPUB22	ETPUB21	ETPUB20	ETPUB19	ETPUB18	ETPUB17	ETPUB16
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

SIU_ETPUBC field descriptions

Field	Description
0 ETPUB15	eTPU_B Channel Select. Select an eTPU_B channel to source the serialized DSPI_C output. 0 This bit in the DSPI_C serialized output frame will not use the respective eTPU_B channel. 1 This bit in the DSPI_C serialized output frame will use the respective eTPU_B channel.
1 ETPUB14	eTPU_B Channel Select. Select an eTPU_B channel to source the serialized DSPI_C output. 0 This bit in the DSPI_C serialized output frame will not use the respective eTPU_B channel. 1 This bit in the DSPI_C serialized output frame will use the respective eTPU_B channel.
2 ETPUB13	eTPU_B Channel Select. Select an eTPU_B channel to source the serialized DSPI_C output. 0 This bit in the DSPI_C serialized output frame will not use the respective eTPU_B channel. 1 This bit in the DSPI_C serialized output frame will use the respective eTPU_B channel.
3 ETPUB12	eTPU_B Channel Select. Select an eTPU_B channel to source the serialized DSPI_C output. 0 This bit in the DSPI_C serialized output frame will not use the respective eTPU_B channel. 1 This bit in the DSPI_C serialized output frame will use the respective eTPU_B channel.
4 ETPUB11	eTPU_B Channel Select. Select an eTPU_B channel to source the serialized DSPI_C output. 0 This bit in the DSPI_C serialized output frame will not use the respective eTPU_B channel. 1 This bit in the DSPI_C serialized output frame will use the respective eTPU_B channel.
5 ETPUB10	eTPU_B Channel Select. Select an eTPU_B channel to source the serialized DSPI_C output. 0 This bit in the DSPI_C serialized output frame will not use the respective eTPU_B channel. 1 This bit in the DSPI_C serialized output frame will use the respective eTPU_B channel.
6 ETPUB9	eTPU_B Channel Select. Select an eTPU_B channel to source the serialized DSPI_C output. 0 This bit in the DSPI_C serialized output frame will not use the respective eTPU_B channel. 1 This bit in the DSPI_C serialized output frame will use the respective eTPU_B channel.
7 ETPUB8	eTPU_B Channel Select. Select an eTPU_B channel to source the serialized DSPI_C output. 0 This bit in the DSPI_C serialized output frame will not use the respective eTPU_B channel. 1 This bit in the DSPI_C serialized output frame will use the respective eTPU_B channel.
8 ETPUB7	eTPU_B Channel Select. Select an eTPU_B channel to source the serialized DSPI_C output. 0 This bit in the DSPI_C serialized output frame will not use the respective eTPU_B channel. 1 This bit in the DSPI_C serialized output frame will use the respective eTPU_B channel.
9 ETPUB6	eTPU_B Channel Select. Select an eTPU_B channel to source the serialized DSPI_C output. 0 This bit in the DSPI_C serialized output frame will not use the respective eTPU_B channel. 1 This bit in the DSPI_C serialized output frame will use the respective eTPU_B channel.

Table continues on the next page...

SIU_ETPUBC field descriptions (continued)

Field	Description
10 ETPUB5	eTPU_B Channel Select. Select an eTPU_B channel to source the serialized DSPI_C output. 0 This bit in the DSPI_C serialized output frame will not use the respective eTPU_B channel. 1 This bit in the DSPI_C serialized output frame will use the respective eTPU_B channel.
11 ETPUB4	eTPU_B Channel Select. Select an eTPU_B channel to source the serialized DSPI_C output. 0 This bit in the DSPI_C serialized output frame will not use the respective eTPU_B channel. 1 This bit in the DSPI_C serialized output frame will use the respective eTPU_B channel.
12 ETPUB3	eTPU_B Channel Select. Select an eTPU_B channel to source the serialized DSPI_C output. 0 This bit in the DSPI_C serialized output frame will not use the respective eTPU_B channel. 1 This bit in the DSPI_C serialized output frame will use the respective eTPU_B channel.
13 ETPUB2	eTPU_B Channel Select. Select an eTPU_B channel to source the serialized DSPI_C output. 0 This bit in the DSPI_C serialized output frame will not use the respective eTPU_B channel. 1 This bit in the DSPI_C serialized output frame will use the respective eTPU_B channel.
14 ETPUB1	eTPU_B Channel Select. Select an eTPU_B channel to source the serialized DSPI_C output. 0 This bit in the DSPI_C serialized output frame will not use the respective eTPU_B channel. 1 This bit in the DSPI_C serialized output frame will use the respective eTPU_B channel.
15 ETPUB0	eTPU_B Channel Select. Select an eTPU_B channel to source the serialized DSPI_C output. 0 This bit in the DSPI_C serialized output frame will not use the respective eTPU_B channel. 1 This bit in the DSPI_C serialized output frame will use the respective eTPU_B channel.
16 ETPUB31	eTPU_B Channel Select. Select an eTPU_B channel to source the serialized DSPI_C output. 0 This bit in the DSPI_C serialized output frame will not use the respective eTPU_B channel. 1 This bit in the DSPI_C serialized output frame will use the respective eTPU_B channel.
17 ETPUB30	eTPU_B Channel Select. Select an eTPU_B channel to source the serialized DSPI_C output. 0 This bit in the DSPI_C serialized output frame will not use the respective eTPU_B channel. 1 This bit in the DSPI_C serialized output frame will use the respective eTPU_B channel.
18 ETPUB29	eTPU_B Channel Select. Select an eTPU_B channel to source the serialized DSPI_C output. 0 This bit in the DSPI_C serialized output frame will not use the respective eTPU_B channel. 1 This bit in the DSPI_C serialized output frame will use the respective eTPU_B channel.
19 ETPUB28	eTPU_B Channel Select. Select an eTPU_B channel to source the serialized DSPI_C output. 0 This bit in the DSPI_C serialized output frame will not use the respective eTPU_B channel. 1 This bit in the DSPI_C serialized output frame will use the respective eTPU_B channel.
20 ETPUB27	eTPU_B Channel Select. Select an eTPU_B channel to source the serialized DSPI_C output. 0 This bit in the DSPI_C serialized output frame will not use the respective eTPU_B channel. 1 This bit in the DSPI_C serialized output frame will use the respective eTPU_B channel.
21 ETPUB26	eTPU_B Channel Select. Select an eTPU_B channel to source the serialized DSPI_C output. 0 This bit in the DSPI_C serialized output frame will not use the respective eTPU_B channel. 1 This bit in the DSPI_C serialized output frame will use the respective eTPU_B channel.

Table continues on the next page...

SIU_ETPUBC field descriptions (continued)

Field	Description
22 ETPUB25	eTPU_B Channel Select. Select an eTPU_B channel to source the serialized DSPI_C output. 0 This bit in the DSPI_C serialized output frame will not use the respective eTPU_B channel. 1 This bit in the DSPI_C serialized output frame will use the respective eTPU_B channel.
23 ETPUB24	eTPU_B Channel Select. Select an eTPU_B channel to source the serialized DSPI_C output. 0 This bit in the DSPI_C serialized output frame will not use the respective eTPU_B channel. 1 This bit in the DSPI_C serialized output frame will use the respective eTPU_B channel.
24 ETPUB23	eTPU_B Channel Select. Select an eTPU_B channel to source the serialized DSPI_C output. 0 This bit in the DSPI_C serialized output frame will not use the respective eTPU_B channel. 1 This bit in the DSPI_C serialized output frame will use the respective eTPU_B channel.
25 ETPUB22	eTPU_B Channel Select. Select an eTPU_B channel to source the serialized DSPI_C output. 0 This bit in the DSPI_C serialized output frame will not use the respective eTPU_B channel. 1 This bit in the DSPI_C serialized output frame will use the respective eTPU_B channel.
26 ETPUB21	eTPU_B Channel Select. Select an eTPU_B channel to source the serialized DSPI_C output. 0 This bit in the DSPI_C serialized output frame will not use the respective eTPU_B channel. 1 This bit in the DSPI_C serialized output frame will use the respective eTPU_B channel.
27 ETPUB20	eTPU_B Channel Select. Select an eTPU_B channel to source the serialized DSPI_C output. 0 This bit in the DSPI_C serialized output frame will not use the respective eTPU_B channel. 1 This bit in the DSPI_C serialized output frame will use the respective eTPU_B channel.
28 ETPUB19	eTPU_B Channel Select. Select an eTPU_B channel to source the serialized DSPI_C output. 0 This bit in the DSPI_C serialized output frame will not use the respective eTPU_B channel. 1 This bit in the DSPI_C serialized output frame will use the respective eTPU_B channel.
29 ETPUB18	eTPU_B Channel Select. Select an eTPU_B channel to source the serialized DSPI_C output. 0 This bit in the DSPI_C serialized output frame will not use the respective eTPU_B channel. 1 This bit in the DSPI_C serialized output frame will use the respective eTPU_B channel.
30 ETPUB17	eTPU_B Channel Select. Select an eTPU_B channel to source the serialized DSPI_C output. 0 This bit in the DSPI_C serialized output frame will not use the respective eTPU_B channel. 1 This bit in the DSPI_C serialized output frame will use the respective eTPU_B channel.
31 ETPUB16	eTPU_B Channel Select. Select an eTPU_B channel to source the serialized DSPI_C output. 0 This bit in the DSPI_C serialized output frame will not use the respective eTPU_B channel. 1 This bit in the DSPI_C serialized output frame will use the respective eTPU_B channel.

8.2.70 Serialized Output Signal Selection Register for DSPI_D (SIU_ETPUBD)

Each bit in the DSPI serialized output frame can contain a signal routed from the output of one of three on-chip sources. The sources on this device are the eTPU module, eMIOS module, or a software-updated GPO data register. The mapping between module or data register and the output frame for a single DSPI channel is performed by independent 32-bit registers, one for each module plus one for the data register.

The following register is used by DSPI_D to select the sources of the serialized output when running in DSI or CSI configuration.

The register bit enables a path from the eTPU_B channel or eMIOS channel to the equivalent bit position in the DSPI_D serialized output frame. The user must ensure that bit selections from each of these registers do not overlap. Multiple sources are logically ORed, which provides the potential for combining outputs from multiple timer channels and data registers to produce more complex bit behavior.

Address: 0h base + D70h offset = D70h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	ETPUB21	ETPUB20	ETPUB19	ETPUB18	ETPUB17	ETPUB16	0				ETPUB29	ETPUB28	ETPUB27	ETPUB26	ETPUB25	ETPUB24
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

SIU_ETPUBD field descriptions

Field	Description
0 ETPUB21	eTPU_B Channel Select. Select an eTPU_B channel to source the serialized DSPI_D output. 0 This bit in the DSPI_D serialized output frame will not use the respective eTPU_B channel. 1 This bit in the DSPI_D serialized output frame will use the respective eTPU_B channel.
1 ETPUB20	eTPU_B Channel Select. Select an eTPU_B channel to source the serialized DSPI_D output. 0 This bit in the DSPI_D serialized output frame will not use the respective eTPU_B channel. 1 This bit in the DSPI_D serialized output frame will use the respective eTPU_B channel.
2 ETPUB19	eTPU_B Channel Select. Select an eTPU_B channel to source the serialized DSPI_D output.

Table continues on the next page...

SIU_ETPUBD field descriptions (continued)

Field	Description
	0 This bit in the DSPI_D serialized output frame will not use the respective eTPU_B channel. 1 This bit in the DSPI_D serialized output frame will use the respective eTPU_B channel.
3 ETPUB18	eTPU_B Channel Select. Select an eTPU_B channel to source the serialized DSPI_D output. 0 This bit in the DSPI_D serialized output frame will not use the respective eTPU_B channel. 1 This bit in the DSPI_D serialized output frame will use the respective eTPU_B channel.
4 ETPUB17	eTPU_B Channel Select. Select an eTPU_B channel to source the serialized DSPI_D output. 0 This bit in the DSPI_D serialized output frame will not use the respective eTPU_B channel. 1 This bit in the DSPI_D serialized output frame will use the respective eTPU_B channel.
5 ETPUB16	eTPU_B Channel Select. Select an eTPU_B channel to source the serialized DSPI_D output. 0 This bit in the DSPI_D serialized output frame will not use the respective eTPU_B channel. 1 This bit in the DSPI_D serialized output frame will use the respective eTPU_B channel.
6–9 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
10 ETPUB29	eTPU_B Channel Select. Select an eTPU_B channel to source the serialized DSPI_D output. 0 This bit in the DSPI_D serialized output frame will not use the respective eTPU_B channel. 1 This bit in the DSPI_D serialized output frame will use the respective eTPU_B channel.
11 ETPUB28	eTPU_B Channel Select. Select an eTPU_B channel to source the serialized DSPI_D output. 0 This bit in the DSPI_D serialized output frame will not use the respective eTPU_B channel. 1 This bit in the DSPI_D serialized output frame will use the respective eTPU_B channel.
12 ETPUB27	eTPU_B Channel Select. Select an eTPU_B channel to source the serialized DSPI_D output. 0 This bit in the DSPI_D serialized output frame will not use the respective eTPU_B channel. 1 This bit in the DSPI_D serialized output frame will use the respective eTPU_B channel.
13 ETPUB26	eTPU_B Channel Select. Select an eTPU_B channel to source the serialized DSPI_D output. 0 This bit in the DSPI_D serialized output frame will not use the respective eTPU_B channel. 1 This bit in the DSPI_D serialized output frame will use the respective eTPU_B channel.
14 ETPUB25	eTPU_B Channel Select. Select an eTPU_B channel to source the serialized DSPI_D output. 0 This bit in the DSPI_D serialized output frame will not use the respective eTPU_B channel. 1 This bit in the DSPI_D serialized output frame will use the respective eTPU_B channel.
15 ETPUB24	eTPU_B Channel Select. Select an eTPU_B channel to source the serialized DSPI_D output. 0 This bit in the DSPI_D serialized output frame will not use the respective eTPU_B channel. 1 This bit in the DSPI_D serialized output frame will use the respective eTPU_B channel.
16–31 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

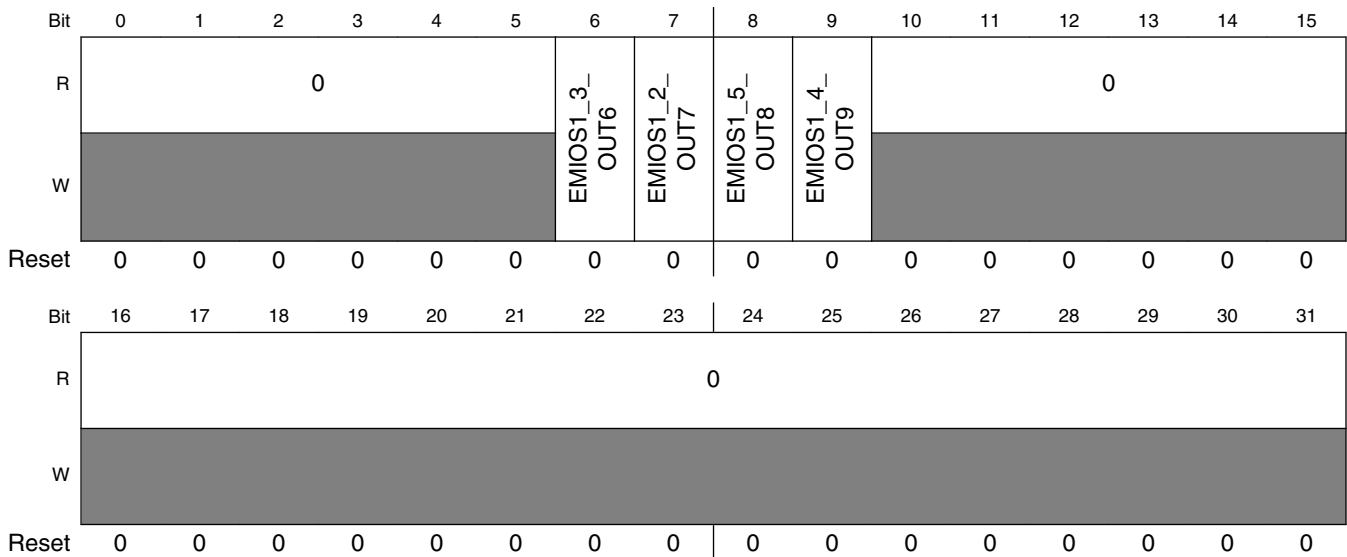
8.2.71 Serialized Output Signal Selection Register for DSPI_D (SIU_EMIOSD)

Each bit in the DSPI serialized output frame can contain a signal routed from the output of one of three on-chip sources. The sources on this device are the eTPU module, eMIOS module, or a software-updated GPO data register. The mapping between module or data register and the output frame for a single DSPI channel is performed by independent 32-bit registers, one for each module plus one for the data register.

The following register is used by DSPI_D to select the sources of the serialized output when running in DSI or CSI configuration.

The register bit enables a path from the eTPU_B channel or eMIOS channel to the equivalent bit position in the DSPI_D serialized output frame. The user must ensure that bit selections from each of these registers do not overlap. Multiple sources are logically ORed, which provides the potential for combining outputs from multiple timer channels and data registers to produce more complex bit behavior.

Address: 0h base + D74h offset = D74h



SIU_EMIOSD field descriptions

Field	Description
0–5 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
6 EMIOS1_3_OUT6	eMIOS_1 Channel Select. Select an eMIOS_1 channel to source the serialized DSPI_D output. 0 This bit in the DSPI_D serialized output frame will not use the respective eMIOS_1 channel 3. 1 This bit in the DSPI_D serialized output frame will use the respective eMIOS_1 channel 3.

Table continues on the next page...

SIU_EMIOSD field descriptions (continued)

Field	Description
7 EMIOS1_2_OUT7	eMIOS_1 Channel Select. Select an eMIOS_1 channel to source the serialized DSPI_D output. 0 This bit in the DSPI_D serialized output frame will not use the respective eMIOS_1 channel 2. 1 This bit in the DSPI_D serialized output frame will use the respective eMIOS_1 channel 2.
8 EMIOS1_5_OUT8	eMIOS_1 Channel Select. Select an eMIOS_1 channel to source the serialized DSPI_D output. 0 This bit in the DSPI_D serialized output frame will not use the respective eMIOS_1 channel 5. 1 This bit in the DSPI_D serialized output frame will use the respective eMIOS_1 channel 5.
9 EMIOS1_4_OUT9	eMIOS_1 Channel Select. Select an eMIOS_1 channel to source the serialized DSPI_D output. 0 This bit in the DSPI_D serialized output frame will not use the respective eMIOS_1 channel 4. 1 This bit in the DSPI_D serialized output frame will use the respective eMIOS_1 channel 4.
10–31 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

8.2.72 Serialized Output Signal Selection Register for DSPI_D (SIU_DSPIDHLD)

Address: 0h base + D78h offset = D78h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	DSPIDH0	DSPIDH1	DSPIDH2	DSPIDH3	DSPIDH4	DSPIDH5	DSPIDH6	DSPIDH7	DSPIDH8	DSPIDH9	DSPIDH10	DSPIDH11	DSPIDH12	DSPIDH13	DSPIDH14	DSPIDH15
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	DSPIDL16	DSPIDL17	DSPIDL18	DSPIDL19	DSPIDL20	DSPIDL21	DSPIDL22	DSPIDL23	DSPIDL24	DSPIDL25	DSPIDL26	DSPIDL27	DSPIDL28	DSPIDL29	DSPIDL30	DSPIDL31
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

SIU_DSPIDHLD field descriptions

Field	Description
0 DSPIDH0	SIU_DSPIDH Select. Select one bit of the DSPI_D GPO Mask Output High Register to source the serialized DSPI_D output. 0 This bit in the DSPI_D serialized output frame will not use the respective SIU_DSPIDH bit. 1 This bit in the DSPI_D serialized output frame will use the respective SIU_DSPIDH bit.
1 DSPIDH1	SIU_DSPIDH Select. Select one bit of the DSPI_D GPO Mask Output High Register to source the serialized DSPI_D output.

Table continues on the next page...

SIU_DSPIDHLD field descriptions (continued)

Field	Description
	0 This bit in the DSPI_D serialized output frame will not use the respective SIU_DSPIDH bit. 1 This bit in the DSPI_D serialized output frame will use the respective SIU_DSPIDH bit.
2 DSPIDH2	SIU_DSPIDH Select. Select one bit of the DSPI_D GPO Mask Output High Register to source the serialized DSPI_D output. 0 This bit in the DSPI_D serialized output frame will not use the respective SIU_DSPIDH bit. 1 This bit in the DSPI_D serialized output frame will use the respective SIU_DSPIDH bit.
3 DSPIDH3	SIU_DSPIDH Select. Select one bit of the DSPI_D GPO Mask Output High Register to source the serialized DSPI_D output. 0 This bit in the DSPI_D serialized output frame will not use the respective SIU_DSPIDH bit. 1 This bit in the DSPI_D serialized output frame will use the respective SIU_DSPIDH bit.
4 DSPIDH4	SIU_DSPIDH Select. Select one bit of the DSPI_D GPO Mask Output High Register to source the serialized DSPI_D output. 0 This bit in the DSPI_D serialized output frame will not use the respective SIU_DSPIDH bit. 1 This bit in the DSPI_D serialized output frame will use the respective SIU_DSPIDH bit.
5 DSPIDH5	SIU_DSPIDH Select. Select one bit of the DSPI_D GPO Mask Output High Register to source the serialized DSPI_D output. 0 This bit in the DSPI_D serialized output frame will not use the respective SIU_DSPIDH bit. 1 This bit in the DSPI_D serialized output frame will use the respective SIU_DSPIDH bit.
6 DSPIDH6	SIU_DSPIDH Select. Select one bit of the DSPI_D GPO Mask Output High Register to source the serialized DSPI_D output. 0 This bit in the DSPI_D serialized output frame will not use the respective SIU_DSPIDH bit. 1 This bit in the DSPI_D serialized output frame will use the respective SIU_DSPIDH bit.
7 DSPIDH7	SIU_DSPIDH Select. Select one bit of the DSPI_D GPO Mask Output High Register to source the serialized DSPI_D output. 0 This bit in the DSPI_D serialized output frame will not use the respective SIU_DSPIDH bit. 1 This bit in the DSPI_D serialized output frame will use the respective SIU_DSPIDH bit.
8 DSPIDH8	SIU_DSPIDH Select. Select one bit of the DSPI_D GPO Mask Output High Register to source the serialized DSPI_D output. 0 This bit in the DSPI_D serialized output frame will not use the respective SIU_DSPIDH bit. 1 This bit in the DSPI_D serialized output frame will use the respective SIU_DSPIDH bit.
9 DSPIDH9	SIU_DSPIDH Select. Select one bit of the DSPI_D GPO Mask Output High Register to source the serialized DSPI_D output. 0 This bit in the DSPI_D serialized output frame will not use the respective SIU_DSPIDH bit. 1 This bit in the DSPI_D serialized output frame will use the respective SIU_DSPIDH bit.
10 DSPIDH10	SIU_DSPIDH Select. Select one bit of the DSPI_D GPO Mask Output High Register to source the serialized DSPI_D output. 0 This bit in the DSPI_D serialized output frame will not use the respective SIU_DSPIDH bit. 1 This bit in the DSPI_D serialized output frame will use the respective SIU_DSPIDH bit.
11 DSPIDH11	SIU_DSPIDH Select. Select one bit of the DSPI_D GPO Mask Output High Register to source the serialized DSPI_D output.

Table continues on the next page...

SIU_DSPIDHLD field descriptions (continued)

Field	Description
	0 This bit in the DSPI_D serialized output frame will not use the respective SIU_DSPIDH bit. 1 This bit in the DSPI_D serialized output frame will use the respective SIU_DSPIDH bit.
12 DSPIDH12	SIU_DSPIDH Select. Select one bit of the DSPI_D GPO Mask Output High Register to source the serialized DSPI_D output. 0 This bit in the DSPI_D serialized output frame will not use the respective SIU_DSPIDH bit. 1 This bit in the DSPI_D serialized output frame will use the respective SIU_DSPIDH bit.
13 DSPIDH13	SIU_DSPIDH Select. Select one bit of the DSPI_D GPO Mask Output High Register to source the serialized DSPI_D output. 0 This bit in the DSPI_D serialized output frame will not use the respective SIU_DSPIDH bit. 1 This bit in the DSPI_D serialized output frame will use the respective SIU_DSPIDH bit.
14 DSPIDH14	SIU_DSPIDH Select. Select one bit of the DSPI_D GPO Mask Output High Register to source the serialized DSPI_D output. 0 This bit in the DSPI_D serialized output frame will not use the respective SIU_DSPIDH bit. 1 This bit in the DSPI_D serialized output frame will use the respective SIU_DSPIDH bit.
15 DSPIDH15	SIU_DSPIDH Select. Select one bit of the DSPI_D GPO Mask Output High Register to source the serialized DSPI_D output. 0 This bit in the DSPI_D serialized output frame will not use the respective SIU_DSPIDH bit. 1 This bit in the DSPI_D serialized output frame will use the respective SIU_DSPIDH bit.
16 DSPIDL16	SIU_DSPIDL Select. Select one bit of the DSPI_D GPO Mask Output Low Register to source the serialized DSPI_D output. 0 This bit in the DSPI_D serialized output frame will not use the respective SIU_DSPIDL bit. 1 This bit in the DSPI_D serialized output frame will use the respective SIU_DSPIDL bit.
17 DSPIDL17	SIU_DSPIDL Select. Select one bit of the DSPI_D GPO Mask Output Low Register to source the serialized DSPI_D output. 0 This bit in the DSPI_D serialized output frame will not use the respective SIU_DSPIDL bit. 1 This bit in the DSPI_D serialized output frame will use the respective SIU_DSPIDL bit.
18 DSPIDL18	SIU_DSPIDL Select. Select one bit of the DSPI_D GPO Mask Output Low Register to source the serialized DSPI_D output. 0 This bit in the DSPI_D serialized output frame will not use the respective SIU_DSPIDL bit. 1 This bit in the DSPI_D serialized output frame will use the respective SIU_DSPIDL bit.
19 DSPIDL19	SIU_DSPIDL Select. Select one bit of the DSPI_D GPO Mask Output Low Register to source the serialized DSPI_D output. 0 This bit in the DSPI_D serialized output frame will not use the respective SIU_DSPIDL bit. 1 This bit in the DSPI_D serialized output frame will use the respective SIU_DSPIDL bit.
20 DSPIDL20	SIU_DSPIDL Select. Select one bit of the DSPI_D GPO Mask Output Low Register to source the serialized DSPI_D output. 0 This bit in the DSPI_D serialized output frame will not use the respective SIU_DSPIDL bit. 1 This bit in the DSPI_D serialized output frame will use the respective SIU_DSPIDL bit.
21 DSPIDL21	SIU_DSPIDL Select. Select one bit of the DSPI_D GPO Mask Output Low Register to source the serialized DSPI_D output.

Table continues on the next page...

SIU_DSPIDHLD field descriptions (continued)

Field	Description
	0 This bit in the DSPI_D serialized output frame will not use the respective SIU_DSPIDL bit. 1 This bit in the DSPI_D serialized output frame will use the respective SIU_DSPIDL bit.
22 DSPIDL22	SIU_DSPIDL Select. Select one bit of the DSPI_D GPO Mask Output Low Register to source the serialized DSPI_D output. 0 This bit in the DSPI_D serialized output frame will not use the respective SIU_DSPIDL bit. 1 This bit in the DSPI_D serialized output frame will use the respective SIU_DSPIDL bit.
23 DSPIDL23	SIU_DSPIDL Select. Select one bit of the DSPI_D GPO Mask Output Low Register to source the serialized DSPI_D output. 0 This bit in the DSPI_D serialized output frame will not use the respective SIU_DSPIDL bit. 1 This bit in the DSPI_D serialized output frame will use the respective SIU_DSPIDL bit.
24 DSPIDL24	SIU_DSPIDL Select. Select one bit of the DSPI_D GPO Mask Output Low Register to source the serialized DSPI_D output. 0 This bit in the DSPI_D serialized output frame will not use the respective SIU_DSPIDL bit. 1 This bit in the DSPI_D serialized output frame will use the respective SIU_DSPIDL bit.
25 DSPIDL25	SIU_DSPIDL Select. Select one bit of the DSPI_D GPO Mask Output Low Register to source the serialized DSPI_D output. 0 This bit in the DSPI_D serialized output frame will not use the respective SIU_DSPIDL bit. 1 This bit in the DSPI_D serialized output frame will use the respective SIU_DSPIDL bit.
26 DSPIDL26	SIU_DSPIDL Select. Select one bit of the DSPI_D GPO Mask Output Low Register to source the serialized DSPI_D output. 0 This bit in the DSPI_D serialized output frame will not use the respective SIU_DSPIDL bit. 1 This bit in the DSPI_D serialized output frame will use the respective SIU_DSPIDL bit.
27 DSPIDL27	SIU_DSPIDL Select. Select one bit of the DSPI_D GPO Mask Output Low Register to source the serialized DSPI_D output. 0 This bit in the DSPI_D serialized output frame will not use the respective SIU_DSPIDL bit. 1 This bit in the DSPI_D serialized output frame will use the respective SIU_DSPIDL bit.
28 DSPIDL28	SIU_DSPIDL Select. Select one bit of the DSPI_D GPO Mask Output Low Register to source the serialized DSPI_D output. 0 This bit in the DSPI_D serialized output frame will not use the respective SIU_DSPIDL bit. 1 This bit in the DSPI_D serialized output frame will use the respective SIU_DSPIDL bit.
29 DSPIDL29	SIU_DSPIDL Select. Select one bit of the DSPI_D GPO Mask Output Low Register to source the serialized DSPI_D output. 0 This bit in the DSPI_D serialized output frame will not use the respective SIU_DSPIDL bit. 1 This bit in the DSPI_D serialized output frame will use the respective SIU_DSPIDL bit.
30 DSPIDL30	SIU_DSPIDL Select. Select one bit of the DSPI_D GPO Mask Output Low Register to source the serialized DSPI_D output. 0 This bit in the DSPI_D serialized output frame will not use the respective SIU_DSPIDL bit. 1 This bit in the DSPI_D serialized output frame will use the respective SIU_DSPIDL bit.
31 DSPIDL31	SIU_DSPIDL Select. Select one bit of the DSPI_D GPO Mask Output Low Register to source the serialized DSPI_D output.

Table continues on the next page...

SIU_DSPIDHLD field descriptions (continued)

Field	Description
0	This bit in the DSPI_D serialized output frame will not use the respective SIU_DSPIDL bit.
1	This bit in the DSPI_D serialized output frame will use the respective SIU_DSPIDL bit.

8.2.73 GPIO Pin Data Input Register (SIU_GPDIn)

Previous devices are limited to supporting a maximum of 256 general purpose input registers, and thus a maximum GPIO input assignment of GPIO[0:255]. The reason was that only 256 bytes of SIU address space were allocated for the GPIO Pin Data Input registers, and adjacent memory locations are unavailable. To accommodate the extended number of the GPIO pads, this device has a new memory map definition of 512 bytes for allocating up to 512 GPIO Pin Data Input registers. For compatibility, the first used 256 GPIO Pin Data Input registers are mapped to both the old and new locations.

Each SIU_GPDIn register specifies the input state for the GPDIn[n] pin. The *n* notation in the SIU_GPDIn register name relates to the *[n]* in GPIO[n] signal name. For example, SIU_GPDI246 contains the PDI246 bit for CNTXD_GPIO246. Gaps exist in the memory map where registers correspond to GPIO pins that are not implemented in the package.

Software reads the SIU_GPDIn registers to get the input state of the external GPIO pin. Each GPDIn register contains the input state of one external GPIO pin. If a GPDIn register is configured as output, and the input buffer enable bit is set to one in the PCR register, the SIU_GPDIn register reflects the state of the output pin.

Table 8-1191. Implemented GPIO Pin Data Input Registers

Offset	GPIO register
0x0E00 - 0x0E4A	Reserved
0x0E4B - 0x0E6E	GPIO Pin Data Input Registers 75-110
0x0E6F - 0x0E70	Reserved
0x0E71 - 0x0ECC	GPIO Pin Data Input Registers 113-204
0x0ECD - 0x0ECF	Reserved
0x0ED0 - 0x0ED5	GPIO Pin Data Input Registers 208-213
0x0ED6 - 0x0EDB	Reserved
0x0EDC - 0x0EDF	GPIO Pin Data Input Registers 220-223
0x0EE0 - 0x0EE6	Reserved
0x0EE7 - 0x0EFD	GPIO Pin Data Input Registers 231-253
0x0EFE - 0x0EFF	Reserved
0x0F00 - 0x0F33	GPIO Pin Data Input Registers 256-307
0x0F34 - 0x0FAF	Reserved
0x0FB0 - 0x0FB5	GPIO Pin Data Input Registers 432-437

Table continues on the next page...

Table 8-1191. Implemented GPIO Pin Data Input Registers (continued)

Offset	GPIO register
0x0FB6 - 0x0FB7	Reserved
0x0FB8 - 0x0FBA	GPI Pin Data Input Registers 440-442
0x0FBB	GPIO Pin Data Register 443
0x0FBC	GPI Pin Data Register 444
0x0FBD - 0x0FC1	GPIO Pin Data Registers 445-449
0x0FC2	GPI Pin Data Register 450
0x0FC3 - 0x0FD8	GPIO Pin Data Registers 451-472
0x0FD9	Reserved
0x0FDA - 0x0FEB	GPI Pin Data Registers 474-491
0x0FEC - 0x0FFF	Reserved

Address: 0h base + E4Bh offset + (1d × i), where i=0d to 416d

Bit	0	1	2	3	4	5	6	7
Read	0						PDIn	
Write								
Reset	0	0	0	0	0	0	0	*

* Notes:

- PDIn field: The value depends on what is being driven on the corresponding pin.

SIU_GPDIn field descriptions

Field	Description
0–6 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
7 PDIn	Pin Data In. This bit reflects the state on the external GPIO/GPI pin. If PCR _n [IBE] is 1, then: 0 Signal on pin is logic zero. 1 Signal on pin is logic one.

8.2.74 Input Multiplexing Register0 (SIU_IMUX0)

The SIU_IMUX registers specify the source of peripheral inputs that may come from multiple package pins. The registers allow selection from 4 sources for each input. Each SIU_IMUX register contains 16 inputs, each with 2 bits (4 sources).

Some registers are reserved for future use, and some fields within some of the registers are also reserved.

Address: 0h base + 1200h offset = 1200h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0								MUXSEL11		MUXSEL10		MUXSEL9		MUXSEL8	
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	MUXSEL7		MUXSEL6		MUXSEL5		MUXSEL4		MUXSEL3		MUXSEL2		MUXSEL1		MUXSEL0	
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

SIU_IMUX0 field descriptions

Field	Description
0–7 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
8–9 MUXSEL11	Selects the source pad for SENT Receiver B Channel 5 Input : SENT5_B. 00 Selects input from "ETPUC10_IRQ1_SENT5_B_GPIO451" pad (PCR: 451). 01 Selects input from "PCSB5_PCSC0_SENT5_B_FEC_RXD2_GPIO110" pad (PCR: 110). 10 Reserved 11 Reserved
10–11 MUXSEL10	Selects the source pad for SENT Receiver B Channel 4 Input : SENT4_B. 00 Selects input from "ETPUC9_IRQ0_SENT4_B_ANB44_GPI450" pad (PCR: 450). 01 Selects input from "PCSB4_SCKC_SENT4_B_FEC_MDC_GPIO109" pad (PCR: 109). 10 Reserved 11 Reserved
12–13 MUXSEL9	Selects the source pad for SENT Receiver B Channel 3 Input : SENT3_B. 00 Selects input from "ETPUC8_PCSE5_SENT3_B_GPIO449" pad (PCR: 449). 01 Selects input from "PCSB3_SINC_SENT3_B_GPIO108" pad (PCR: 108). 10 Reserved 11 Reserved
14–15 MUXSEL8	Selects the source pad for SENT Receiver B Channel 2 Input : SENT2_B. 00 Selects input from "ETPUC7_PCSE4_SENT2_B_GPIO448" pad (PCR: 448). 01 Selects input from "PCSB2_SOUTC_SENT2_B_FEC_RXD3_GPIO107" pad (PCR: 107). 10 Reserved 11 Reserved
16–17 MUXSEL7	Selects the source pad for SENT Receiver B Channel 1 Input : SENT1_B. 00 Selects input from "ETPUC6_PCSE3_SENT1_B_GPIO447" pad (PCR: 447). 01 Selects input from "SCKA_PCSC1_MCANTX0_SENT1_B_GPIO93" pad (PCR: 93). 10 Reserved 11 Reserved
18–19 MUXSEL6	Selects the source pad for SENT Receiver B Channel 0 Input : SENT0_B. 00 Selects input from "ETPUC5_PCSE2_SENT0_B_GPIO446" pad (PCR: 446). 01 Selects input from "FEC_TXCLK_REFCLK_SENT0_B_GPI474" pad (PCR: 474). 10 Reserved 11 Reserved

Table continues on the next page...

SIU_IMUX0 field descriptions (continued)

Field	Description
20–21 MUXSEL5	Selects the source pad for SENT Receiver A Channel 5 Input : SENT5_A. 00 Selects input from "ETPUC4_SENT5_A_TCRCLKC_GPIO445" pad (PCR: 445). 01 Selects input from "FEC_RXD1_SENT5_A_GPIO253" pad (PCR: 253). 10 Reserved 11 Reserved
22–23 MUXSEL4	Selects the source pad for SENT Receiver A Channel 4 Input : SENT4_A. 00 Selects input from "ETPUC3_SENT4_A_ANB45_GPI444" pad (PCR: 444). 01 Selects input from "FEC_TX_EN_SENT4_A_GPIO252" pad (PCR: 252). 10 Reserved 11 Reserved
24–25 MUXSEL3	Selects the source pad for SENT Receiver A Channel 3 Input : SENT3_A. 00 Selects input from "ETPUC2_SENT3_A_GPIO443" pad (PCR: 443). 01 Selects input from "FEC_TXD1_SENT3_A_GPIO251" pad (PCR: 251). 10 Reserved 11 Reserved
26–27 MUXSEL2	Selects the source pad for SENT Receiver A Channel 2 Input : SENT2_A. 00 Selects input from "SENT2_A_ANB43_GPI475" pad (PCR: 475). 01 Selects input from "FEC_RXD0_SENT2_A_GPIO250" pad (PCR: 250). 10 Reserved 11 Reserved
28–29 MUXSEL1	Selects the source pad for SENT Receiver A Channel 1 Input : SENT1_A. 00 Selects input from "ETPUC1_SENT1_A_ANB42_GPI442" pad (PCR: 442). 01 Selects input from "FEC_RX_DV_SENT1_A_GPIO249" pad (PCR: 249). 10 Reserved 11 Reserved
30–31 MUXSEL0	Selects the source pad for SENT Receiver A Channel 0 Input : SENT0_A. 00 Selects input from "ETPUC0_SENT0_A_ANB41_GPI441" pad (PCR: 441). 01 Selects input from "FEC_TXD0_SENT0_A_GPIO248" pad (PCR: 248). 10 Reserved 11 Reserved

8.2.75 Input Multiplexing Register1 (SIU_IMUX1)

The SIU_IMUX registers specify the source of peripheral inputs that may come from multiple package pins. The registers allow selection from 4 sources for each input. Each SIU_IMUX register contains 16 inputs, each with 2 bits (4 sources).

Some registers are reserved for future use, and some fields within some of the registers are also reserved.

Address: 0h base + 1204h offset = 1204h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0				MUXSEL5		0						MUXSEL1		MUXSEL0	
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

SIU_IMUX1 field descriptions

Field	Description
0–19 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
20–21 MUXSEL5	Selects the source pad for FlexCAN D receive : CNRXD. 00 Selects input from "CNRXD_MCANRX1_GPIO247" pad (PCR: 247). 01 Selects input from "EMIOS15_IRQ1_CNRXD_RCH1_C_GPIO194" pad (PCR: 194). 10 Reserved 11 Reserved
22–27 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
28–29 MUXSEL1	Selects the source pad for MCAN 1 Rx data : MCANRX1. 00 Selects input from "ETPUC15_MCANRX1_RCH2_C_GPIO456" pad (PCR: 456). 01 Selects input from "CNRXD_MCANRX1_GPIO247" pad (PCR: 247). 10 Selects input from "EMIOS17_ETPUB1_MCANRX1_RCH3_C_GPIO196" pad (PCR: 196). 11 Reserved
30–31 MUXSEL0	Selects the source pad for MCAN 0 Rx data : MCANRX0. 00 Selects input from "ETPUA19_PCSD4_MCANRX0_GPIO133" pad (PCR: 133). 01 Selects input from "CNRXA_RXDA_MCANRX0_GPIO84" pad (PCR: 84). 10 Selects input from "TXDB_PCSD1_MCANRX0_GPIO91" pad (PCR: 91). 11 Selects input from "PCSC5_MCANRX0_SOUT_D_LVDISM_PCSD0_GPIO243" pad (PCR: 243).

8.2.76 Input Multiplexing Register2 (SIU_IMUX2)

The SIU_IMUX registers specify the source of peripheral inputs that may come from multiple package pins. The registers allow selection from 4 sources for each input. Each SIU_IMUX register contains 16 inputs, each with 2 bits (4 sources).

Some registers are reserved for future use, and some fields within some of the registers are also reserved.

Memory Map and Register Definition

Address: 0h base + 1208h offset = 1208h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0								MUXSEL11		MUXSEL10		MUXSEL9		MUXSEL8	
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	MUXSEL7		MUXSEL6		MUXSEL5		0									
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

SIU_IMUX2 field descriptions

Field	Description
0–7 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
8–9 MUXSEL11	Selects the source pad for DSPI D peripheral chip select 0 : PCSD0. 00 Selects input from "PCSB1_PCSD0_GPIO106" pad (PCR: 106). 01 Selects input from "ETPUC28_PCSD0_RCH6_C_MAB2_GPIO469" pad (PCR: 469). 10 Selects input from "PCSC5_MCANRX0_SOUT_D_LVDSM_PCSD0_GPIO243" pad (PCR: 243). 11 Reserved
10–11 MUXSEL10	Selects the source pad for DSPI D data input : SIND. 00 Selects input from "PCSC3_SCK_D_LVDSM_SIND_GPIO241" pad (PCR: 241). 01 Selects input from "ETPUC31_SIND_RCH9_C_GPIO472" pad (PCR: 472). 10 Selects input from "EMIOS11_SIND_GPIO190" pad (PCR: 190). 11 Reserved
12–13 MUXSEL9	Selects the source pad for DSPI D clock : SCKD. 00 Selects input from "PCSC2_SCK_D_LVDSP_SCKD_GPIO240" pad (PCR: 240). 01 Selects input from "ETPUC29_SCKD_RCH7_C_GPIO470" pad (PCR: 470). 10 Selects input from "EMIOS10_SCKD_GPIO189" pad (PCR: 189). 11 Reserved
14–15 MUXSEL8	Selects the source pad for DSPI C peripheral chip select 0 : PCSC0. 00 Selects input from "PCSC0_SOUT_C_LVDSM_GPIO238" pad (PCR: 238). 01 Selects input from "PCSB5_PCSC0_SENT5_B_FEC_RXD2_GPIO110" pad (PCR: 110). 10 Selects input from "EMIOS28_PCSC0_GPIO434" pad (PCR: 434). 11 Reserved
16–17 MUXSEL7	Selects the source pad for DSPI C data input : SINC. 00 Selects input from "SINC_SCK_C_LVDSM_GPIO236" pad (PCR: 236). 01 Selects input from "PCSB3_SINC_SENT3_B_GPIO108" pad (PCR: 108). 10 Reserved 11 Reserved
18–19 MUXSEL6	Selects the source pad for DSPI C clock : SCKC. 00 Selects input from "SCKC_SCK_C_LVDSP_GPIO235" pad (PCR: 235). 01 Selects input from "PCSB4_SCKC_SENT4_B_FEC_MDC_GPIO109" pad (PCR: 109). 10 Reserved 11 Reserved

Table continues on the next page...

SIU_IMUX2 field descriptions (continued)

Field	Description
20–21 MUXSEL5	Selects the source pad for DSPI B peripheral chip select 0 : PCSB0. 00 Selects input from "PCSB0_PCSD2_GPIO105" pad (PCR: 105). 01 Selects input from "EMIOS24_PCSB0_GPIO203" pad (PCR: 203). 10 Reserved 11 Reserved
22–31 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

8.2.77 Input Multiplexing Register3 (SIU_IMUX3)

The SIU_IMUX registers specify the source of peripheral inputs that may come from multiple package pins. The registers allow selection from 4 sources for each input. Each SIU_IMUX register contains 16 inputs, each with 2 bits (4 sources).

Some registers are reserved for future use, and some fields within some of the registers are also reserved

Address: 0h base + 120Ch offset = 120Ch

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0		MUXSEL6		MUXSEL5		MUXSEL4		MUXSEL3		MUXSEL2		MUXSEL1		MUXSEL0	
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

SIU_IMUX3 field descriptions

Field	Description
0–17 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
18–19 MUXSEL6	Selects the source pad for External interrupt request 6 : IRQ6. 00 Selects input from "TCRCLKB_IRQ6_GPIO146" pad (PCR: 146). 01 Selects input from "TCRCLKC_IRQ6_ANB40_GPI440" pad (PCR: 440). 10 Reserved 11 Reserved
20–21 MUXSEL5	Selects the source pad for External interrupt request 5 : IRQ5. 00 Selects input from "PLLCFG1_IRQ5_SOUTD_SIPI_TXP_GPIO209" pad (PCR: 209). 01 Selects input from "ETPUC14_IRQ5_SDIN0_PSI5A_RCH1_C_GPIO455" pad (PCR: 455).

Table continues on the next page...

SIU_IMUX3 field descriptions (continued)

Field	Description
	10 Reserved 11 Reserved
22–23 MUXSEL4	Selects the source pad for External interrupt request 4 : IRQ4. 00 Selects input from "PLLCFG0_IRQ4_ETPUC9_SIPI_TXN_GPIO208" pad (PCR: 208). 01 Selects input from "ETPUC13_IRQ4_SDOUT0_PSI5A_RCH0_C_GPIO454" pad (PCR: 454). 10 Reserved 11 Reserved
24–25 MUXSEL3	Selects the source pad for External interrupt request 3 : IRQ3. 00 Selects input from "BOOTCFG1_IRQ3_ETPUC0_SIPI_RXP_GPIO212" pad (PCR: 212). 01 Selects input from "ETPUC12_IRQ3_MCANTX1_GPIO453" pad (PCR: 453). 10 Reserved 11 Reserved
26–27 MUXSEL2	Selects the source pad for External interrupt request 2 : IRQ2. 00 Selects input from "BOOTCFG0_IRQ2_GPIO211" pad (PCR: 211). 01 Selects input from "ETPUC11_IRQ2_GPIO452" pad (PCR: 452). 10 Reserved 11 Reserved
28–29 MUXSEL1	Selects the source pad for External interrupt request 1 : IRQ1. 00 Selects input from "EMIOS15_IRQ1_CN_RXD_RCH1_C_GPIO194" pad (PCR: 194). 01 Selects input from "ETPUC10_IRQ1_SENT5_B_GPIO451" pad (PCR: 451). 10 Reserved 11 Reserved
30–31 MUXSEL0	Selects the source pad for External interrupt request 0 : IRQ0. 00 Selects input from "EMIOS14_IRQ0_CNTXD_RCH0_C_GPIO193" pad (PCR: 193). 01 Selects input from "ETPUC9_IRQ0_SENT4_B_ANB44_GPI450" pad (PCR: 450). 10 Reserved 11 Reserved

8.2.78 Input Multiplexing Register4 (SIU_IMUX4)

The SIU_IMUX registers specify the source of peripheral inputs that may come from multiple package pins. The registers allow selection from 4 sources for each input. Each SIU_IMUX register contains 16 inputs, each with 2 bits (4 sources).

Some registers are reserved for future use, and some fields within some of the registers are also reserved

Address: 0h base + 1210h offset = 1210h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0														MUXSEL8	
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0								MUXSEL3		MUXSEL2		MUXSEL1		MUXSEL0	
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

SIU_IMUX4 field descriptions

Field	Description
0–13 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
14–15 MUXSEL8	Selects the source pad for eSCI E transmit : TXDE. 00 Selects input from "PCSA4_SCKE_TXDE_FEC_RXCLK_GPIO100" pad (PCR: 100). 01 Selects input from "MDO3_TXDE_GPIO223" pad (PCR: 223). 10 Reserved 11 Reserved
16–23 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
24–25 MUXSEL3	Selects the source pad for eSCI B transmit : TXDB. 00 Selects input from "TXDB_PCSD1_MCANRX0_GPIO91" pad (PCR: 91). 01 Selects input from "ETPUC21_TXDB_GPIO462" pad (PCR: 462). 10 Reserved 11 Reserved
26–27 MUXSEL2	Selects the source pad for eSCI B receive : RXDB. 00 Selects input from "RXDB_PCSD5_GPIO92" pad (PCR: 92). 01 Selects input from "ETPUC22_RXDB_GPIO463" pad (PCR: 463). 10 Reserved 11 Reserved
28–29 MUXSEL1	Selects the source pad for eSCI A receive : RXDA. 00 Selects input from "RXDA_GPIO90" pad (PCR: 90). 01 Selects input from "CNRXA_RXDA_MCANRX0_GPIO84" pad (PCR: 84). 10 Selects input from "ETPUC20_RXDA_SDIN0_PSI5B_GPIO461" pad (PCR: 461). 11 Reserved
30–31 MUXSEL0	Selects the source pad for eSCI A transmit : TXDA. 00 Selects input from "TXDA_GPIO89" pad (PCR: 89). 01 Selects input from "CNTXA_TXDA_MCANTX0_GPIO83" pad (PCR: 83). 10 Selects input from "ETPUC19_TXDA_SDOUT0_PSI5B_GPIO460" pad (PCR: 460). 11 Reserved

8.2.79 Input Multiplexing Register5 (SIU_IMUX5)

The SIU_IMUX registers specify the source of peripheral inputs that may come from multiple package pins. The registers allow selection from 4 sources for each input. Each SIU_IMUX register contains 16 inputs, each with 2 bits (4 sources).

Some registers are reserved for future use, and some fields within some of the registers are also reserved

Address: 0h base + 1214h offset = 1214h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R																
W																
	MUXSEL15		MUXSEL14		MUXSEL13		MUXSEL12		MUXSEL11		MUXSEL10		MUXSEL9		MUXSEL8	
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R																
W																
	MUXSEL7		MUXSEL6		MUXSEL5		MUXSEL4		MUXSEL3		MUXSEL2		MUXSEL1		MUXSEL0	
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

SIU_IMUX5 field descriptions

Field	Description
0–1 MUXSEL15	Selects the source pad for eMIOS 1 channel 23 : EMIOS31. 00 Selects input from "EMIOS31_PCSC5_GPIO437" pad (PCR: 437). 01 Selects input from "ANB23_ETPUC8_EMIOS31_GPI491" pad (PCR: 491). 10 Reserved 11 Reserved
2–3 MUXSEL14	Selects the source pad for eMIOS 1 channel 22 : EMIOS30. 00 Selects input from "EMIOS30_PCSC2_GPIO436" pad (PCR: 436). 01 Selects input from "ANB22_ETPUC9_EMIOS30_GPI490" pad (PCR: 490). 10 Reserved 11 Reserved
4–5 MUXSEL13	Selects the source pad for eMIOS 1 channel 21 : EMIOS29. 00 Selects input from "EMIOS29_PCSC1_GPIO435" pad (PCR: 435). 01 Selects input from "ANB21_ETPUC10_EMIOS29_GPI489" pad (PCR: 489). 10 Reserved 11 Reserved
6–7 MUXSEL12	Selects the source pad for eMIOS 1 channel 20 : EMIOS28. 00 Selects input from "EMIOS28_PCSC0_GPIO434" pad (PCR: 434). 01 Selects input from "ANB20_ETPUC11_EMIOS28_GPI488" pad (PCR: 488). 10 Reserved 11 Reserved
8–9 MUXSEL11	Selects the source pad for eMIOS 1 channel 19 : EMIOS27. 00 Selects input from "EMIOS27_PCSB3_GPIO433" pad (PCR: 433).

Table continues on the next page...

SIU_IMUX5 field descriptions (continued)

Field	Description
	01 Selects input from "ANB19_ETPUC12_EMIO527_GPI487" pad (PCR: 487). 10 Reserved 11 Reserved
10–11 MUXSEL10	Selects the source pad for eMIOS 1 channel 18 : EMIO526. 00 Selects input from "EMIO526_PCSB2_GPIO432" pad (PCR: 432). 01 Selects input from "ANB18_ETPUC13_EMIO526_GPI486" pad (PCR: 486). 10 Reserved 11 Reserved
12–13 MUXSEL9	Selects the source pad for eMIOS 1 channel 17 : EMIO525. 00 Selects input from "EMIO525_PCSB1_GPIO204" pad (PCR: 204). 01 Selects input from "ANB17_ETPUC14_EMIO525_GPI485" pad (PCR: 485). 10 Reserved 11 Reserved
14–15 MUXSEL8	Selects the source pad for eMIOS 1 channel 16 : EMIO524. 00 Selects input from "EMIO524_PCSB0_GPIO203" pad (PCR: 203). 01 Selects input from "ANB16_ETPUC15_EMIO524_GPI484" pad (PCR: 484). 10 Reserved 11 Reserved
16–17 MUXSEL7	Selects the source pad for eMIOS 0 channel 23 : EMIO523. 00 Selects input from "EMIO523_ETPUB7_RCH9_C_GPIO202" pad (PCR: 202). 01 Selects input from "ANB15_ETPUA24_EMIO523_GPI483" pad (PCR: 483). 10 Reserved 11 Reserved
18–19 MUXSEL6	Selects the source pad for eMIOS 0 channel 22 : EMIO522. 00 Selects input from "EMIO522_ETPUB6_RCH8_C_GPIO201" pad (PCR: 201). 01 Selects input from "ANB14_ETPUA25_EMIO522_GPI482" pad (PCR: 482). 10 Reserved 11 Reserved
20–21 MUXSEL5	Selects the source pad for eMIOS 0 channel 21 : EMIO521. 00 Selects input from "EMIO521_ETPUB5_RCH7_C_GPIO200" pad (PCR: 200). 01 Selects input from "ANB13_ETPUA26_EMIO521_GPI481" pad (PCR: 481). 10 Reserved 11 Reserved
22–23 MUXSEL4	Selects the source pad for eMIOS 0 channel 20 : EMIO520. 00 Selects input from "EMIO520_ETPUB4_RCH6_C_GPIO199" pad (PCR: 199). 01 Selects input from "ANB12_ETPUA27_EMIO520_GPI480" pad (PCR: 480). 10 Reserved 11 Reserved
24–25 MUXSEL3	Selects the source pad for eMIOS 0 channel 19 : EMIO519. 00 Selects input from "EMIO519_ETPUB3_RCH5_C_GPIO198" pad (PCR: 198).

Table continues on the next page...

SIU_IMUX5 field descriptions (continued)

Field	Description
	01 Selects input from "ANB11_ETPUA28_EMIOS19_GPI479" pad (PCR: 479). 10 Reserved 11 Reserved
26–27 MUXSEL2	Selects the source pad for eMIOS 0 channel 18 : EMIOS18. 00 Selects input from "EMIOS18_ETPUB2_RCH4_C_GPIO197" pad (PCR: 197). 01 Selects input from "ANB10_ETPUA29_EMIOS18_GPI478" pad (PCR: 478). 10 Reserved 11 Reserved
28–29 MUXSEL1	Selects the source pad for eMIOS 0 channel 17 : EMIOS17. 00 Selects input from "EMIOS17_ETPUB1_MCANRX1_RCH3_C_GPIO196" pad (PCR: 196). 01 Selects input from "ANB9_ETPUA30_EMIOS17_GPI477" pad (PCR: 477). 10 Reserved 11 Reserved
30–31 MUXSEL0	Selects the source pad for eMIOS 0 channel 16 : EMIOS16. 00 Selects input from "EMIOS16_ETPUB0_MCANTX1_RCH2_C_GPIO195" pad (PCR: 195). 01 Selects input from "ANB8_ETPUA31_EMIOS16_GPI476" pad (PCR: 476). 10 Reserved 11 Reserved

8.2.80 Input Multiplexing Register7 (SIU_IMUX7)

The SIU_IMUX registers specify the source of peripheral inputs that may come from multiple package pins. The registers allow selection from 4 sources for each input. Each SIU_IMUX register contains 16 inputs, each with 2 bits (4 sources).

Some registers are reserved for future use, and some fields within some of the registers are also reserved.

Address: 0h base + 121Ch offset = 121Ch

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R																
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

SIU_IMUX7 field descriptions

Field	Description
0–1 MUXSEL15	Selects the source pad for eTPU A channel 31 : ETPUA31. 00 Selects input from "ETPUA31_PCSC4_RCH9_A_GPIO145" pad (PCR: 145). 01 Selects input from "ANB8_ETPUA31_EMIO16_GPI476" pad (PCR: 476). 10 Reserved 11 Reserved
2–3 MUXSEL14	Selects the source pad for eTPU A channel 30 : ETPUA30. 00 Selects input from "ETPUA30_PCSC3_RCH8_A_GPIO144" pad (PCR: 144). 01 Selects input from "ANB9_ETPUA30_EMIO17_GPI477" pad (PCR: 477). 10 Reserved 11 Reserved
4–5 MUXSEL13	Selects the source pad for eTPU A channel 29 : ETPUA29. 00 Selects input from "ETPUA29_PCSC2_RCH7_A_GPIO143" pad (PCR: 143). 01 Selects input from "ANB10_ETPUA29_EMIO18_GPI478" pad (PCR: 478). 10 Reserved 11 Reserved
6–7 MUXSEL12	Selects the source pad for eTPU A channel 28 : ETPUA28. 00 Selects input from "ETPUA28_PCSC1_RCH6_A_GPIO142" pad (PCR: 142). 01 Selects input from "ANB11_ETPUA28_EMIO19_GPI479" pad (PCR: 479). 10 Reserved 11 Reserved
8–9 MUXSEL11	Selects the source pad for eTPU A channel 27 : ETPUA27. 00 Selects input from "ETPUA27_IRQ15_RCH5_A_GPIO141" pad (PCR: 141). 01 Selects input from "ANB12_ETPUA27_EMIO20_GPI480" pad (PCR: 480). 10 Reserved 11 Reserved
10–11 MUXSEL10	Selects the source pad for eTPU A channel 26 : ETPUA26. 00 Selects input from "ETPUA26_IRQ14_RCH4_A_GPIO140" pad (PCR: 140). 01 Selects input from "ANB13_ETPUA26_EMIO21_GPI481" pad (PCR: 481). 10 Reserved 11 Reserved
12–13 MUXSEL9	Selects the source pad for eTPU A channel 25 : ETPUA25. 00 Selects input from "ETPUA25_IRQ13_RCH3_A_GPIO139" pad (PCR: 139). 01 Selects input from "ANB14_ETPUA25_EMIO22_GPI482" pad (PCR: 482). 10 Reserved 11 Reserved
14–15 MUXSEL8	Selects the source pad for eTPU A channel 24 : ETPUA24. 00 Selects input from "ETPUA24_IRQ12_RCH2_A_GPIO138" pad (PCR: 138). 01 Selects input from "ANB15_ETPUA24_EMIO23_GPI483" pad (PCR: 483). 10 Reserved 11 Reserved

Table continues on the next page...

SIU_IMUX7 field descriptions (continued)

Field	Description
16–31 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

8.2.81 Input Multiplexing Register10 (SIU_IMUX10)

The SIU_IMUX registers specify the source of peripheral inputs that may come from multiple package pins. The registers allow selection from 4 sources for each input. Each SIU_IMUX register contains 16 inputs, each with 2 bits (4 sources).

Some registers are reserved for future use, and some fields within some of the registers are also reserved.

Address: 0h base + 1228h offset = 1228h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R																
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R																
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

SIU_IMUX10 field descriptions

Field	Description
0–1 MUXSEL15	Selects the source pad for eTPU C channel 15 : ETPUC15. 00 Selects input from "ETPUC15_MCANRX1_RCH2_C_GPIO456" pad (PCR: 456). 01 Selects input from "ANB16_ETPUC15_EMIO24_GPI484" pad (PCR: 484). 10 Reserved 11 Reserved
2–3 MUXSEL14	Selects the source pad for eTPU C channel 14 : ETPUC14. 00 Selects input from "ETPUC14_IRQ5_SDIN0_PSI5A_RCH1_C_GPIO455" pad (PCR: 455). 01 Selects input from "ANB17_ETPUC14_EMIO25_GPI485" pad (PCR: 485). 10 Reserved 11 Reserved
4–5 MUXSEL13	Selects the source pad for eTPU C channel 13 : ETPUC13. 00 Selects input from "ETPUC13_IRQ4_SDOUT0_PSI5A_RCH0_C_GPIO454" pad (PCR: 454). 01 Selects input from "ANB18_ETPUC13_EMIO26_GPI486" pad (PCR: 486). 10 Reserved 11 Reserved

Table continues on the next page...

SIU_IMUX10 field descriptions (continued)

Field	Description
6–7 MUXSEL12	Selects the source pad for eTPU C channel 12 : ETPUC12. 00 Selects input from "ETPUC12_IRQ3_MCANTX1_GPIO453" pad (PCR: 453). 01 Selects input from "ANB19_ETPUC12_EMIO27_GPIO487" pad (PCR: 487). 10 Reserved 11 Reserved
8–9 MUXSEL11	Selects the source pad for eTPU C channel 11 : ETPUC11. 00 Selects input from "ETPUC11_IRQ2_GPIO452" pad (PCR: 452). 01 Selects input from "ANB20_ETPUC11_EMIO28_GPIO488" pad (PCR: 488). 10 Reserved 11 Reserved
10–11 MUXSEL10	Selects the source pad for eTPU C channel 10 : ETPUC10. 00 Selects input from "ETPUC10_IRQ1_SENT5_B_GPIO451" pad (PCR: 451). 01 Selects input from "ANB21_ETPUC10_EMIO29_GPIO489" pad (PCR: 489). 10 Reserved 11 Reserved
12–13 MUXSEL9	Selects the source pad for eTPU C channel 9 : ETPUC9. 00 Selects input from "ETPUC9_IRQ0_SENT4_B_ANB44_GPIO450" pad (PCR: 450). 01 Selects input from "ANB22_ETPUC9_EMIO30_GPIO490" pad (PCR: 490). 10 Selects input from "PLLCFG0_IRQ4_ETPUC9_SIPI_TXN_GPIO208" pad (PCR: 208). 11 Reserved
14–15 MUXSEL8	Selects the source pad for eTPU C channel 8 : ETPUC8. 00 Selects input from "ETPUC8_PCSE5_SENT3_B_GPIO449" pad (PCR: 449). 01 Selects input from "ANB23_ETPUC8_EMIO31_GPIO491" pad (PCR: 491). 10 Reserved 11 Reserved
16–23 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
24–25 MUXSEL3	Selects the source pad for eTPU C channel 3 : ETPUC3. 00 Selects input from "ETPUC3_SENT4_A_ANB45_GPIO444" pad (PCR: 444). 01 Selects input from "PLLCFG2_ETPUC3_PCSE1_SIPI_REFCLK_GPIO210" pad (PCR: 210). NOTE: The PLLCFG2 function is only effective during reset. After reset, the pin takes other alternate functions, and these other functions are the ones intended to be selected by this mux. 10 Reserved 11 Reserved
26–27 Reserved	This field is reserved.
28–29 MUXSEL1	Selects the source pad for eTPU C channel 1 : ETPUC1. 00 Selects input from "ETPUC1_SENT1_A_ANB42_GPIO442" pad (PCR: 442). 01 Selects input from "WKPCFG_NMI_ETPUC1_SIPI_RXN_GPIO213" pad (PCR: 213).

Table continues on the next page...

SIU_IMUX10 field descriptions (continued)

Field	Description
	NOTE: The WKPCFG function is only effective during reset. After reset, the pin takes other alternate functions, and these other functions are the ones intended to be selected by this mux. 10 Reserved 11 Reserved
30–31 MUXSEL0	Selects the source pad for eTPU C channel 0 : ETPUC0. 00 Selects input from "ETPUC0_SENT0_A_ANB41_GPI441" pad (PCR: 441). 01 Selects input from "BOOTCFG1_IRQ3_ETPUC0_SIPI_RXP_GPIO212" pad (PCR: 212). NOTE: The BOOTCFG1 function is only effective during reset. After reset, the pin takes other alternate functions, and these other functions are the ones intended to be selected by this mux. 10 Reserved 11 Reserved

8.2.82 Input Multiplexing Register12 (SIU_IMUX12)

The SIU_IMUX registers specify the source of peripheral inputs that may come from multiple package pins. The registers allow selection from 4 sources for each input. Each SIU_IMUX register contains 16 inputs, each with 2 bits (4 sources).

Some registers are reserved for future use, and some fields within some of the registers are also reserved.

Address: 0h base + 1230h offset = 1230h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0								MUXSEL2				0			
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

SIU_IMUX12 field descriptions

Field	Description
0–25 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
26–27 MUXSEL2	Selects the source pad for eTPU C TCR clock : TCRCLKC. 00 Selects input from "TCRCLKC_IRQ6_ANB40_GPI440" pad (PCR: 440). 01 Selects input from "ETPUC4_SENT5_A_TCRCLKC_GPIO445" pad (PCR: 445).

Table continues on the next page...

SIU_IMUX12 field descriptions (continued)

Field	Description
10	Reserved
11	Reserved
28–31 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

8.3 SIU_B

SIU_B contains a set of mirrored PCR configuration registers, GPO data registers, and GPI data registers for GPIO[240:243] balls.

By default after reset, the mirrored registers are disabled and provide no function. The disabled state is indicated by PA[2:0] = 111b in the mirrored PCR (PCR_{nM}) register.

When a mirrored PCR_{nM} is enabled by writing a different value to its PA[2:0]:

- The PCR_{nM} has priority over the equivalent non-mirrored PCR register.
- The PCR_{nM} controls the ball function and characteristics defined by its implemented bit fields.
- The mirrored GPO data (GPDO_{nM}) and GPI data (GPDIn_M) registers control and reflect the logic level.

When the mirrored registers are enabled, writes to the non-mirrored registers have no effect.

Offsets for SIU_B registers are relative to SIU_B's base address, which differs from SIU's base address.

SIU_B memory map

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
220	Pad Configuration Register Mirrored (SIU_B_PCR240M)	16	R/W	See section	8.3.1/424
222	Pad Configuration Register Mirrored (SIU_B_PCR241M)	16	R/W	See section	8.3.1/424
224	Pad Configuration Register Mirrored (SIU_B_PCR242M)	16	R/W	See section	8.3.1/424
226	Pad Configuration Register Mirrored (SIU_B_PCR243M)	16	R/W	See section	8.3.1/424
6F0	GPIO Pin Data Output Register Mirrored (SIU_B_GPDO240M)	8	R/W	00h	8.3.2/427

Table continues on the next page...

SIU_B memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
6F1	GPIO Pin Data Output Register Mirrored (SIU_B_GPDO241M)	8	R/W	00h	8.3.2/427
6F2	GPIO Pin Data Output Register Mirrored (SIU_B_GPDO242M)	8	R/W	00h	8.3.2/427
6F3	GPIO Pin Data Output Register Mirrored (SIU_B_GPDO243M)	8	R/W	00h	8.3.2/427
8F0	GPIO Pin Data Input Register Mirrored (SIU_B_GPDI240M)	8	R	00h	8.3.3/428
8F1	GPIO Pin Data Input Register Mirrored (SIU_B_GPDI241M)	8	R	00h	8.3.3/428
8F2	GPIO Pin Data Input Register Mirrored (SIU_B_GPDI242M)	8	R	00h	8.3.3/428
8F3	GPIO Pin Data Input Register Mirrored (SIU_B_GPDI243M)	8	R	00h	8.3.3/428

8.3.1 Pad Configuration Register Mirrored (SIU_B_PCRnM)

By default after reset, each mirrored register is disabled and provides no function. The disabled state is indicated by the reset value of PA[2:0] = 111b.

When a mirrored PCR_{nM} is enabled by writing a different value to its PA[2:0]:

- The PCR_{nM} has priority over the equivalent non-mirrored PCR register.
- The PCR_{nM} controls the ball function and characteristics defined by its implemented bit fields.

When the mirrored registers are enabled, writes to the non-mirrored registers have no effect.

These 16-bit registers are read from or written to as:

- 16-bit values aligned on 16-bit boundaries, or
- 32-bit values aligned on 32-bit address boundaries.

The PCRs allow software to control the following range of electrical characteristics of the external pads:

- Weak pullup/down enable/disable
- Weak pullup/down selection
- Slew-rate selection for outputs
- Drive strength selection for outputs
- Input buffer enable (when direction is configured for output)
- Input hysteresis enable/disable
- Open drain/push-pull output selection
- Multiplexed function selection
- Data direction selection

NOTE

The actual characteristics available in any SIU_B_PCR n M depend on the type of pad it controls. See the SIU_B_PCR n M definition table.

All device pin names begin with the primary function, followed by the alternate function, and then GPIO. See [External signal descriptions, multiplexing, and attributes](#) for how to identify the source module for primary and alternate functions and the description of these signals. For an additional signal description, see the chapter for the module that uses the signal.

There are a number of input signals on the device that have multiple external pins as input sources. Only one input source can be active at any time for these pins. To achieve this, a set of multiplexers controlled by the SIU_IMUX registers select which input will drive the function. See [Input Multiplexing Register0 \(SIU_IMUX0\)](#) to [Input Multiplexing Register12 \(SIU_IMUX12\)](#) for more details.

[Table 8-2037](#) defines the symbols for PCR n M fields' reset values that appear in [Table 8-2038](#).

Table 8-2037. SIU_B_PCR n M field reset-value symbols

Symbol	Meaning
—	Not implemented in the PCR n M. Software should write a default value of 0.
0 or 1	Function is implemented. For the exact definition of the 0 or 1 reset value, see the applicable field's description.

Table 8-2038. SIU_B_PCR[240:243]M reset values

PCR <i>n</i> M	Offset (hex)	PCR <i>n</i> M field reset values												
		PA2	PA1	PA0	OBE	IBE	DSC1	DSC0	ODE	HYS	SRC1	SRC0	WPE	WPS
PCR240M	220	1	1	1	0	0	—	—	0	0	0	0	1	1
PCR241M	222	1	1	1	0	0	—	—	0	0	0	0	1	1
PCR242M	224	1	1	1	0	0	—	—	0	0	0	0	1	1
PCR243M	226	1	1	1	0	0	—	—	0	0	0	0	1	1

Address: 0h base + 220h offset + (2d × i), where i=0d to 3d

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Read	0			PA			OBE	IBE	DSC		ODE	HYS	SRC		WPE	WPS
Write																
Reset	0	0	0	1	1	1	0	0	*	*	0	0	0	0	1	1

* Notes:

- DSC field: This function is not implemented. Software should write a default value of 0 to each bit position.

SIU_B_PCRnM field descriptions

Field	Description
0–2 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
3–5 PA	Pin assignment. Selects the function of a multiplexed pad. A separate port enable output signal from the SIU is asserted for each value of this register. The size of the field can be from 1 to 3 bits, depending on the amount of multiplexing on the pad. 000 GPIO. 001 Primary function. See the description of the corresponding SIU_PCRn. 010 Alternate 1. See the description of the corresponding SIU_PCRn. 011 Alternate 2. See the description of the corresponding SIU_PCRn. 100 Alternate 3. See the description of the corresponding SIU_PCRn. 101 Invalid value. 110 Invalid value. 111 SIU_B's mirrored PCRnM, GPDO _n M, and GPDInM registers are disabled and provide no function.
6 OBE	Output buffer enable. Enables the pad as an output and drives the output buffer enable signal. 0 Disable output buffer for the pad. 1 Enable output buffer for the pad.
7 IBE	Input buffer enable. Enables the pad as an input and drives the input buffer enable signal. 0 Disable input buffer for the pad. 1 Enable input buffer for the pad.
8–9 DSC	Drive strength control. Controls the pad drive strength. 00 10 pF drive strength. 01 20 pF drive strength. 10 30 pF drive strength. 11 50 pF drive strength.
10 ODE	Open drain output enable. Controls output driver configuration for the pads. Either open drain or push/pull driver configurations can be selected. This feature applies to output pins only. 0 Disable open drain for the pad (push/pull driver enabled). 1 Enable open drain for the pad.
11 HYS	Input hysteresis. Controls whether hysteresis is enabled for the pad. 0 Disable hysteresis for the pad. 1 Enable hysteresis for the pad.
12–13 SRC	Slew rate control. Controls slew rate for the pad. Slew rate control pertains to pins with slow or medium I/O pad types, and the output signals are driven according to the value of this field. Actual slew rate depends on the pad type and load. Refer to the electrical specifications for this information. 00 Minimum slew rate. 01 Medium slew rate. 10 Invalid value. 11 Maximum slew rate.
14 WPE	Weak pull up/down enable. Controls whether the weak pullup/down devices are enabled/disabled for the pad. Pullup/down devices are enabled by default.

Table continues on the next page...

SIU_B_PCRnM field descriptions (continued)

Field	Description
	0 Disable weak pull device for the pad. 1 Enable weak pull device for the pad.
15 WPS	Weak pull up/down select. Controls whether weak pullup or weak pulldown devices are used for the pad when weak pullup/down devices are enabled. During reset, the weak pull up/down behavior is controlled by the WKPCFG bit of the Miscellaneous DCF client, or by the WKPCFG pin when operating in legacy mode ((LEG=1 in the Miscellaneous DCF client). The WPS bit determines whether weak pullup or pulldown devices are used after reset, or for pads in which the WKPCFG pin does not determine the reset weak pullup/down state. 0 Pulldown is enabled for the pad. 1 Pullup is enabled for the pad.

8.3.2 GPIO Pin Data Output Register Mirrored (SIU_B_GPDO_nM)

By default after reset, each mirrored register is disabled and provides no function. The disabled state is indicated by PA[2:0] = 111b in the mirrored PCR (PCR_nM) register.

When a mirrored PCR_nM is enabled by writing a different value to its PA[2:0], the mirrored GPO data (GPDO_nM) and GPI data (GPDInM) registers control and reflect the logic level.

When the mirrored registers are enabled, writes to the non-mirrored registers have no effect.

Each GPDO register specifies the output data for the function assigned to the GPIO[*n*] pin. The *n* notation in the SIU_B_GPDO_nM register name relate to the [*n*] in GPIO[*n*] signal name. For example, SIU_B_GPDO240M contains the PDO240 bit for CNTXD_GPIO240.

Software writes to the SIU_B_GPDO_nM registers to drive data out on the external pin. Each register drives one external pin, which allows independent control of the pin. Writes to the SIU_B_GPDO_nM register are not reflected to the pin when a primary or alternate function is assigned or when an input function is assigned to the pin.

If the direction of a GPIO signal changes from input to output, the SIU_B_GPDO_nM register value is automatically driven out to the external pin without the need for a software update.

Writes to a SIU_B_GPDO_nM register have no effect when a primary or alternate function is assigned.

SIU_B

Address: 0h base + 6F0h offset + (1d × i), where i=0d to 3d

Bit	0	1	2	3	4	5	6	7
Read	0						PDOn	
Write								
Reset	0	0	0	0	0	0	0	0

SIU_B_GPDO*n*M field descriptions

Field	Description
0–6 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
7 PDOn	Pin Data Out. This bit stores the data to be driven out on the external GPIO. If the register is read, it returns the value written. 0 Logic low value is driven to the external GPIO pin when the pin is configured as an output. 1 Logic high value is driven to the external GPIO pin when the pin is configured as an output.

8.3.3 GPIO Pin Data Input Register Mirrored (SIU_B_GPDInM)

By default after reset, each mirrored register is disabled and provides no function. The disabled state is indicated by PA[2:0] = 111b in the mirrored PCR (PCR*n*M) register.

When a mirrored PCR*n*M is enabled by writing a different value to its PA[2:0], the mirrored GPO data (GPDO*n*M) and GPI data (GPDInM) registers control and reflect the logic level.

When the mirrored registers are enabled, writes to the non-mirrored registers have no effect.

Each SIU_B_GPDInM register specifies the input state for the GPDIn[n] pin. The *n* notation in the SIU_GPDIn register name relates to the [n] in GPIO[n] signal name. For example, SIU_B_GPDI240M contains the PDI240 bit for CNTXD_GPIO240.

Software reads the SIU_B_GPDInM registers to get the input state of the external GPIO pin. Each GPDIn register contains the input state of one external GPIO pin. If a GPDIn register is configured as output, and the input buffer enable bit is set to 1 in the PCR*n*M register, the SIU_B_GPDInM register reflects the state of the output pin.

Address: 0h base + 8F0h offset + (1d × i), where i=0d to 3d

Bit	0	1	2	3	4	5	6	7
Read	0						PDIn	
Write								
Reset	0	0	0	0	0	0	0	0

SIU_B_GPDInM field descriptions

Field	Description
0–6 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
7 PDIn	Pin Data In. This bit reflects the state on the external GPIO/GPI pin. If PCRnM[IBE] is 1, then: 0 Signal on pin is logic zero. 1 Signal on pin is logic one.

8.4 Functional Description

The following sections provide a functional overview of the SIU operation.

8.4.1 Pad configuration

The pad configuration registers (SIU_PCR) in the SIU allow software control over the following electrical characteristics of the external pads:

- Weak pullup/down enable/disable
- Weak pullup/down selection
- Slew-rate selection for outputs
- Drive strength selection for outputs
- Input buffer enable (when direction is configured for output)
- Input hysteresis enable/disable
- Open drain/push-pull output selection
- Multiplexed function selection
- Data direction selection

The pad configuration registers are provided to allow centralized control over external pins that are shared by more than one module. Each pad configuration register controls a single pin.

8.4.2 Interrupts

There are 16 external interrupt inputs $\overline{\text{IRQ}}[0]–\overline{\text{IRQ}}[15]$ to the SIU. The IRQ inputs can be configured for rising-edge events, falling-edge events, or both.

External interrupt requests are triggered by rising- and/or falling-edge events that are enabled by setting a bit in:

- IRQ rising-edge event enable register (SIU_IREER)
- IRQ falling-edge event enable register (SIU_IFEER)

If the bit is set in both registers, both rising- and falling-edge events trigger an interrupt request. Each IRQ has a counter that tracks the number of system clock cycles that occur between the rising- and falling-edge events. An IRQ counter exists for each IRQ rising- or falling-edge event enable bit.

The digital filter length field in the IRQ digital filter register (SIU_IDFR) specifies the minimum number of system clocks that the IRQ signal must hold a logic value to qualify the edge-triggered event as a valid state change. When the number of system clocks in the IRQ counter equals the value in the digital filter length field, the IRQ state latches and the IRQ counter is cleared.

If the previous filtered state of the IRQ does not match the current state, and the rising- or falling-edge event is enabled, the IRQ flag bit is set to 1. For example, the IRQ flag bit is set if a rising-edge event occurs under the following conditions:

- Previous filtered IRQ state was a logic 0
- Current latched IRQ state is a logic 1
- Rising-edge event is enabled for the IRQ

When the counter for an IRQ is not enabled, the state of the IRQ is held in the current and previous state latches. The IRQ counter operates independently of the IRQ or overrun flag bit. Clearing the IRQ flag or overrun flag bits does not clear or reload the counter.

8.4.2.1 External interrupts

The IRQ signals map to 16 independent interrupt requests output from the SIU. The IRQ flag bit is set when a rising-edge and/or falling-edge event occurs for the IRQ. An external IRQ signal is asserted when all of the following occur:

- Enable bit is set in the IRQ rising- and/or falling-edge event registers (SIU_IREER, SIU_IFEER)
- IRQ flag bit is set in the external interrupt status register (SIU_EISR)
- Enable bit is set in the DMA/Interrupt request enable register (SIU_DIRER)
- Select bit is cleared in the DMA/Interrupt select register (SIU_DIRSR)

The NMI pin function or platform SWT can generate either an NMI or a critical interrupt. FCCU may also generate NMI if any fault is configured to cause NMI and this fault is triggered.

When WKPCFG_NMI_GPIO213 is enabled as NMI, the pin will override the PCR configuration after reset. SIU_DIRER selects between critical and non maskable interrupt use, SIU_EISR reports status of NMI and SIU_IFEER selects edge sensitivity of NMI input. See in the figure below the NMI and critical interrupt connections.

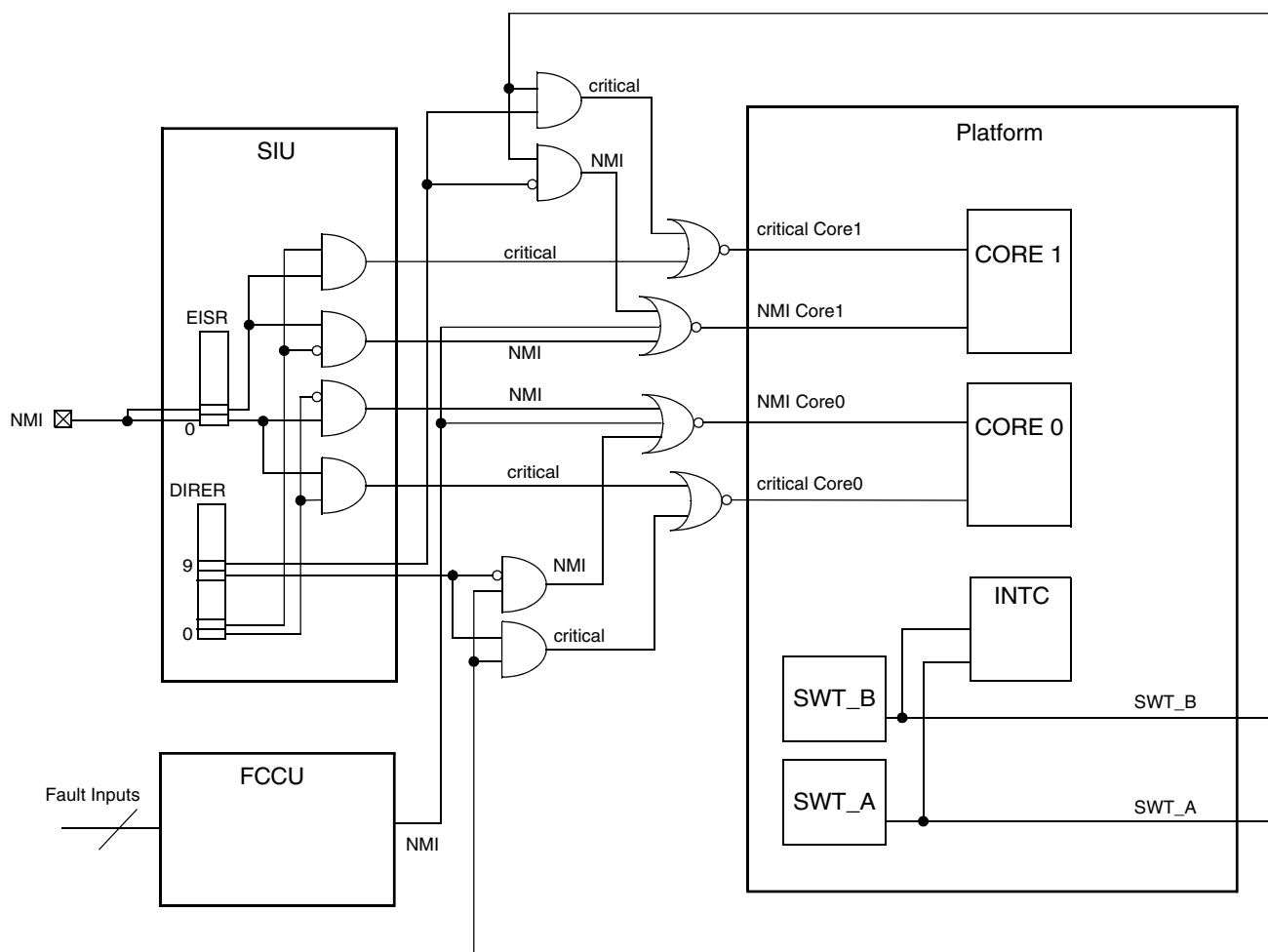


Figure 8-1613. NMI, critical, and SWT interrupts

8.4.2.2 DMA transfers

DMA IRQ signals ($\overline{\text{IRQ}}[0]$ through $\overline{\text{IRQ}}[3]$) map to four independent DMA transfer or interrupt request outputs configured in the SIU. A DMA transfer or interrupt request asserts when all of the following occur:

- IRQ flag bit is set in the external interrupt status register (SIU_EISR)
- Enable bit is set in the DMA transfer or interrupt request enable register (SIU_DIRER)
- Select bit is set in the DMA transfer or interrupt request select register (SIU_DIRSR)

The SIU receives a “DMA transfer done” signal for each DMA or interrupt request transmitted. When the “DMA done” signal asserts, the IRQ flag bit is cleared.

8.4.2.3 Overruns

An overrun IRQ exists for each overrun flag bit in the overrun status register (SIU_OSR). An overrun IRQ asserts when all of the following occur:

- Enable bit is set in the IRQ rising- and/or falling-edge event registers (SIU_IREER, SIU_IFEER)
- IRQ flag bit is set in the external interrupt status register (SIU_EISR)
- Bit is set in the overrun request enable and overrun status registers (SIU_ORER, SIU_OSR)
- Rising- or falling-edge event triggers an interrupt request

The SIU outputs one overrun IRQ bit that is the logical OR of all of the IRQ overrun bits.

8.4.2.4 Edge-detect events

An IRQ asserts when:

- An edge-detect event is enabled
- An edge-detect event occurs

To assert an IRQ when an edge-detect event occurs:

1. Set the enable bit in the IRQ rising- and falling-edge event enable registers (SIU_IREER, SIU_IFEER)
2. Set the enable bits for the DMA/Interrupt request enable register (SIU_DIRER)

The IRQ bit is set in the external IRQ status register (SIU_EISR) when an edge-detect event occurs for that IRQ. The following figure illustrates the behavior of DMA/interrupt requests within the SIU.

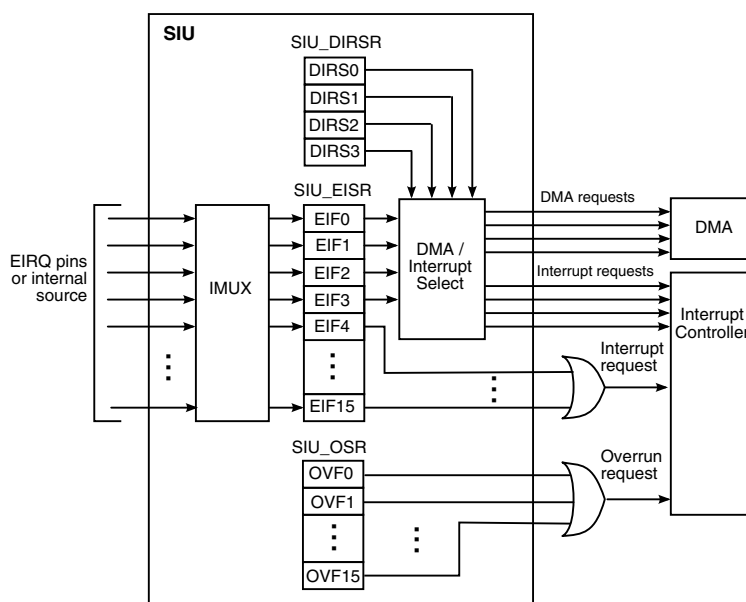


Figure 8-1614. SIU DMA/interrupt request diagram

8.4.3 General Purpose I/O (GPIO)

All GPIO functions for the device are provided by the SIU. Each device pad that has a GPIO signal has a pin configuration register (PCR) in the SIU where the GPIO function is selected. In addition, each device GPIO signal has an input data register (SIU_GPDI n) and an output data register (SIU_GPDO n).

The SIU also implements several parallel GPIO registers (SIU_PGPDO n and SIU_PGPDIn) that can be used to access up to 32 GPIO bits in a single- and word-sized accesses. The values read/written to these parallel registers are coherent with the data read/written to the SIU_GPDO n and SIU_GPDI n registers.

8.4.4 Pin serialization for DSPI

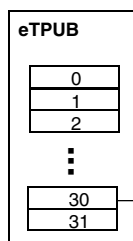
The SIU has four 32-bit outputs to DSPI modules. Each data frame contains the result of the selection, bit by bit, of three sources of data. Each data frame could receive data from eTPU, eMIOS, or an internally implemented GPO register. These registers are defined to provide compliance with the AutoSAR requirements for a PORT driver and support for coherent bit manipulation of up to 16 serialized data bits in a multitasking or multi-processor system. Because the DSPI frame size has been extended to a maximum of 32 bits, the coherent bit manipulation mechanism is partitioned into two separate 16-bit halves of the frame. The purpose of these registers is to allow any combination of bits in each half of the 32-bit serialized data frame to be updated with a single 32-bit write

operation, while allowing other bits to maintain their previous state. This is accomplished by writing a 16-bit masked value coherently with an update value contained in a 16 bit output field, and only updating those bits in the output register for which the corresponding mask bit is set.

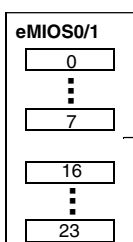
The selection of which bits are actually output on the DSPI data frame is determined by a separate signal selection register. Each register bit enables a path from a single eTPU channel, eMIOS channel, and masked GPO register bit to the equivalent bit position in the DSPI channel frame. Note that each DSPI bit position can be driven by multiple inputs. Software must ensure that the correct bit sources are enabled for correct insertion of the required bit in the bit frame. Multiple sources are logically ORed, which provides the potential for combining outputs from multiple timer channels and data registers to produce more complex bit behavior. An example of how these registers are used to create the DSPI serialized output is shown below.

eTPU Select Register (for DSPI_A)

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
Chan #	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Enable	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

**eMIOS Select Register (for DSPI_A)**

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
Chan #	0_7	0_6	0_5	0_4	0_3	0_2	0_1	0_0	1_0	1_1	1_2	1_3	1_4	1_5	1_6	1_7	0_16	0_17	0_18	0_19	0_20	0_21	0_22	0_23	0_0	0_1	0_2	0_3	0_4	0_5	0_6	0_7
Enable	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

**DSPIH/L Select Register (for DSPI_A)**

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

SIU_DSPIL[MASK17]

SIU_DSPIL[DATA17]

Serial
GPO A**DSPIA Output Register**

Frame bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
Function & Channel Selection																																

Figure 8-1615. Example of pin multiplexing

8.4.5 Internal multiplexing

The internal multiplexing select registers (SIU_ISEL4–8, SIU_EIISR, and SIU_DISR) select the input sources for the following components:

- eTPUA[24:29] inputs
- eQADC command FIFO trigger sources
- SIU external interrupt request signals

8.4.5.1 eQADC external trigger input multiplexing

The eQADC external trigger inputs can connect to one of the following pins:

- External pin
- eTPU channel
- eMIOS channel
- Periodic interrupt timer

The input source for each eQADC external trigger is configured in the eQADC Command FIFO Trigger Source Select Registers (SIU_ISEL4–7). For example, you can select one of the following signals to source the ETRIG[0] input trigger for the eQADC:

- TXDC_ETRIG[0]_GPIO[244] pin
- eTPUA[30] channel output
- eMIOS_1 channel 2 output
- Extended trigger source selection

All ETRIG inputs are multiplexed in the same manner. If an eTPU or eMIOS channel is selected as an ETRIG input to the eQADC, you can activate the alternate function of that eTPU or eMIOS signal on the external pin.

8.4.6 Trigger Burst Generators (TBG)

The Trigger Burst Generator module generates triggers for the eQADC CFIFOs based on a trigger window provided by several sources connected to the IMUX. eTPU, eMIOS, PIT and external SoC inputs can be selected by the IMUX, thus allowed to provide a time window in which the triggers are generated by the Trigger Burst Generator (TBG) module. The TBG is in bypass out of reset, thus backwards compatible functionality is provided. In this backwards compatible setting, the triggers are generated by the signals connected to the IMUX, thus TBG is transparent to the trigger generation. The following figure shows the Trigger Burst Generator block diagram and the waveforms associated with the input trigger window, the output trigger burst (on the left) and the counter waveform (on the right).

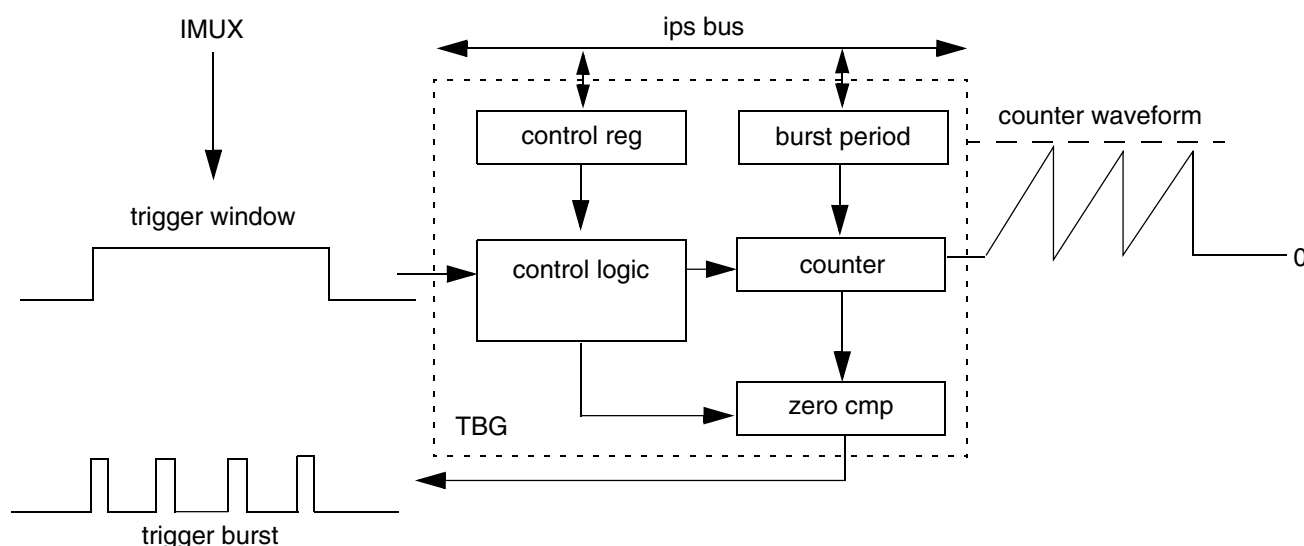


Figure 8-1616. Trigger Burst Generator block diagram

The following figure is a block diagram of the Trigger Burst Generator module interconnection with other modules. The TBG is connected between the multiplexers controlled by ISEL4–7 and eQADC_A/b CIFIFO trigger inputs. There are six CFIFOs on each eQADC and two instances of the eQADC modules, thus 12 TBG instances are used in order to provide trigger burst for all CFIFOs.

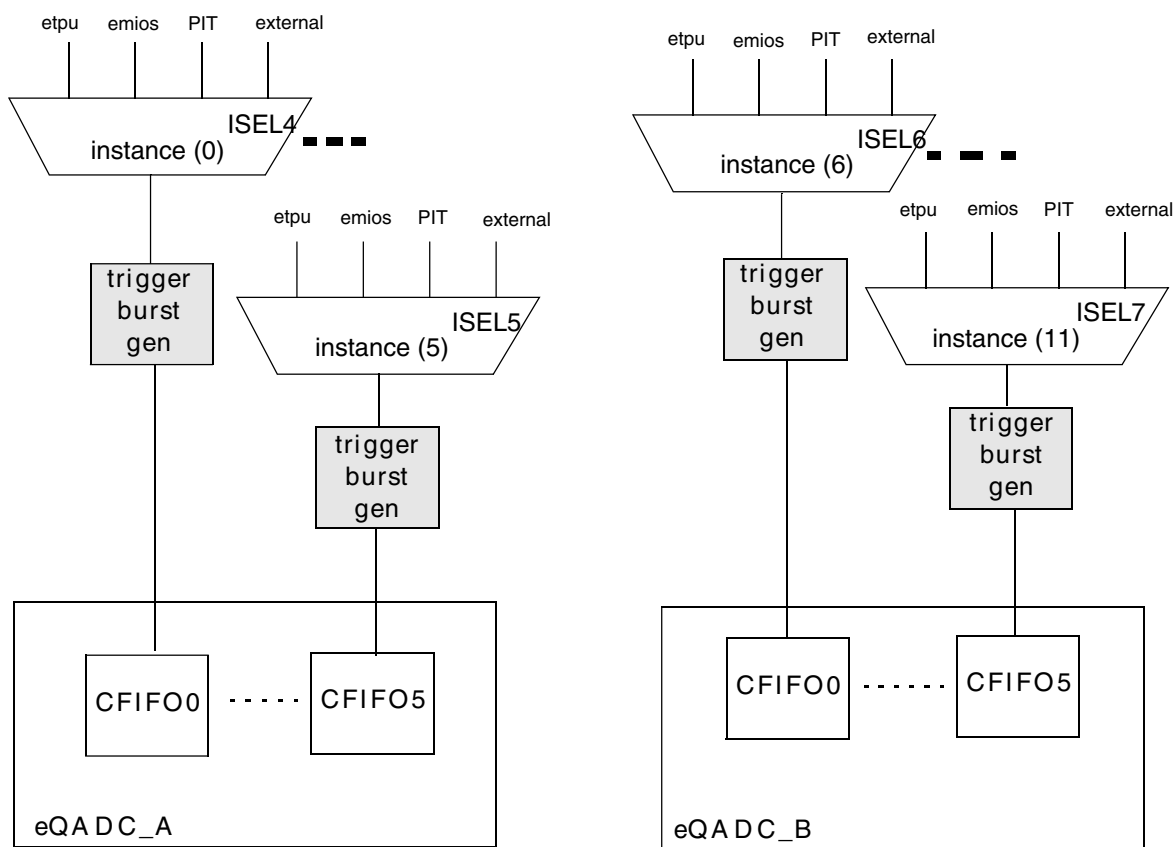


Figure 8-1617. Trigger Burst Generator interconnection

The TBG instances generate a continuous sequence of triggers while the signals selected by ISEL4–7 are asserted. When the trigger sources deassert, then the trigger burst is stopped. The output is identified as the trigger window in the subsequent sections. The opening of a trigger window means that the ISEL4–7 mux outputs enable the TBGs to start generating triggers.

8.4.6.1 Clock selection for the trigger burst

There are three clocks that can be selected for the TBG module. The system clock, the ADC0 clock, and the ADC1 clock. If no clock is selected the output trigger remains negated. It is necessary to have GEN = 0 when changing the CLKSEL to avoid unwanted triggers.

Figure 8-1618 shows the timing diagram of the signals when the peripheral clock (per_clk) is selected to generate the trigger burst. In this case, the trigger counter is enabled to count in all per_clk pulses. Therefore, the time interval between two trigger pulses is calculated as $(\text{TRIGPER} + 1)$ per_clk periods. The figure shows the counter values when TRIGPER is configured to 4.

The first trigger occurs after one clock cycle after the trigger window begins. And the last trigger pulse occurs before the trigger window ends. Therefore, there is no trigger pulse outside the trigger window.

NOTE

When the trigger period register is zero ($\text{TRIGPER} = 0$), the trigger output switches to ON one per_clk period after the trigger window is opened, and remains ON (no transition) until the trigger window is closed.

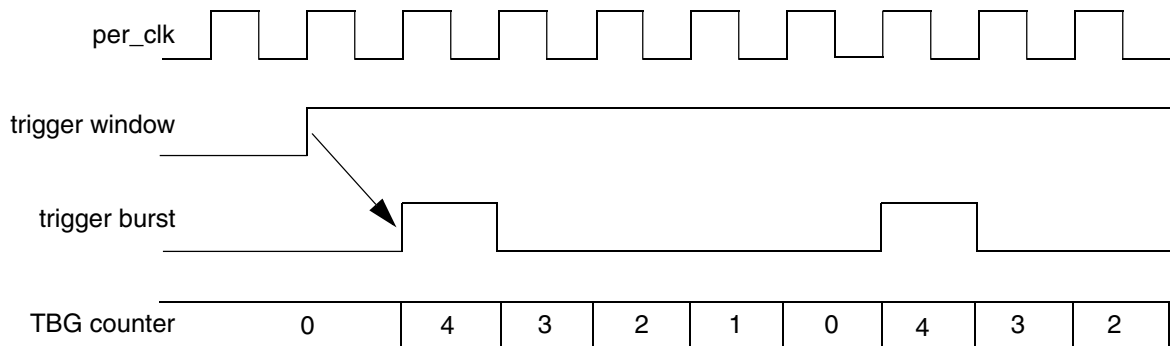


Figure 8-1618. Trigger Burst Generator timing diagram when per_clk is selected

Figure 8-1619 shows the timing diagram of the signals when the ADC0/1 clocks are selected to generate the trigger burst. In this case, the trigger counter is enabled to count only in the positive edge events of that ADC clock. Therefore, the time interval between two trigger pulses is calculated as $(\text{TRIGPER} + 1)$ ADC clock periods.

NOTE

When the trigger period register is zero ($\text{TRIGPER} = 0$) the trigger output pulse occurs for each ADC clock pulse.

The first trigger occurs one system clock cycle after the first detected rising edge of the selected ADC clock after the opening of the trigger window. And the last trigger pulse occurs before the trigger window closes. Therefore, there is no trigger pulse outside the trigger window.

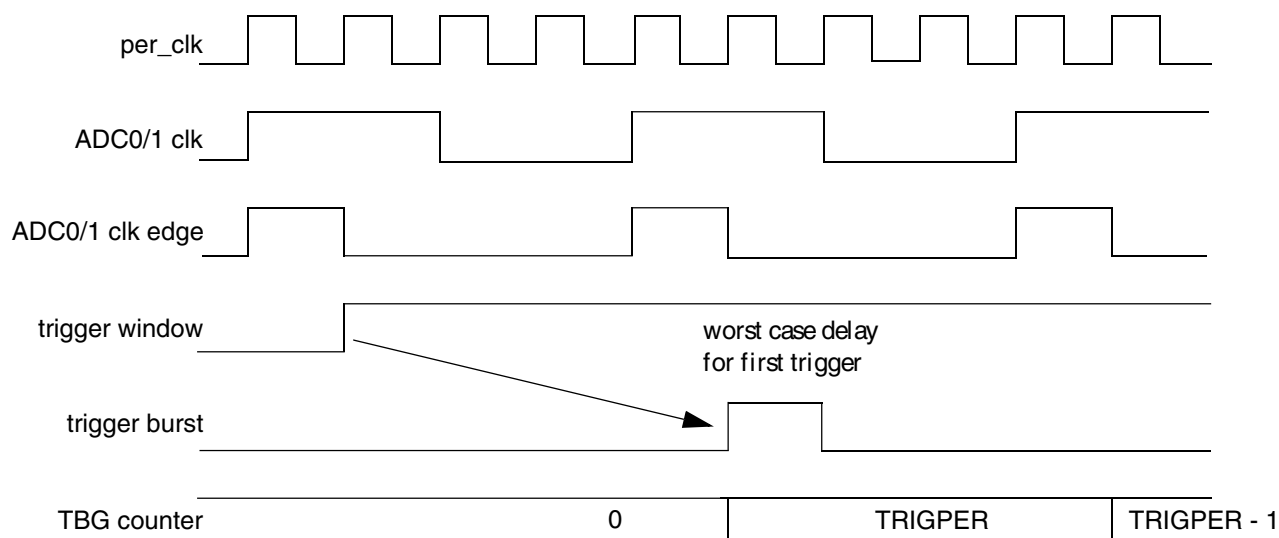


Figure 8-1619. Trigger Burst Generator timing diagram when ADC0/1 is selected

8.4.6.2 Trigger generator configuration procedure

The trigger burst generator (TBG) is disabled out of reset because the GEN bit at the SIU_TBGR_CR_A/Bn register from reset state is clear. This means the trigger window and internal/external selector input signals are bypassed to the respective output ports.

It is recommended to disable the TBG by clearing the GEN bit when the trigger period has to be changed. However, the input trigger window signal at the input port is bypassed to the trigger burst output. If this unwanted trigger must be avoided, the GEN bit must be maintained set and the CLKSEL bit must be configured to all 0 (the trigger counter is cleared at this time). Then it is possible to safely change the trigger period.

Chapter 9

Crossbar Switch (XBAR)

9.1 Chip-specific XBAR information

This chip has one instance of the XBAR module.

9.1.1 XBAR master and slave assignments

The following table lists the XBAR physical port numbers and logical IDs for all master ports on this SoC.

- Each port number matches the default priority assigned to the corresponding physical master port. This default priority equals the reset value of the priority field for each master port in the PRS n registers.
- A priority value of 0 is the highest priority. There is no "disabled" value for the priority.
- A Nexus_3 module and core data bus share the same physical master port for each core.

The logical master ID corresponds to the logical address provided by the master module and is unique for each module. The logical master IDs are used by the bus masters connected to the XBAR. The Nexus master is identified by setting the MSB in the 4-bit field that supplies the master ID number.

Table 9-1. XBAR master ports and logical master IDs

Module	Physical master port	Logical master ID	Comment
Core0 instruction	0	0	
Core0 data	1	0	
Nexus_3_0		8	Nexus_3_0 arbitrates with Core0 data for XBAR port 1
Core1 instruction	2	1	
Core1 data	3	1	
Nexus_3_1		9	Nexus_3_1 arbitrates with Core1 data for XBAR port 3

Table continues on the next page...

Table 9-1. XBAR master ports and logical master IDs (continued)

Module	Physical master port	Logical master ID	Comment
eDMA_A	4	4	
eDMA_B	5	5	
CSE	6	3	CSE and SIPI are connected to master 6 port through a concentrator
SIPI		6	
FEC	7	2	FEC on dedicated master via 2:1 clock gasket

The following table lists the XBAR slave port assignments and the master ports that can access each slave port. Each of the two slave ports for the Flash Memory Controller (PFLASH) module is dedicated to a different core master port. The two slave ports for SRAM support concurrent data accesses to the SRAM.

Table 9-2. XBAR slave port assignments

Module	Slave port	Select Address	Accessible by master port numbers
PFLASH	0	Address[31:29] = 000b	0,1 ¹
EBI	1	Address[31:29] = 001b	0,1,2,3,4,5,6,7
PRAMC_0	2	Address[31:18] = 0100_0000_0000_00b	0,1,2,3,4,5,6,7
PFLASH	3	Address[31:29] = 000b	2,3,4,5,6,7
PRAMC_1	4	Address[31:18] = 0100_0000_0000_01b	0,1,2,3,4,5,6,7
—	5	—	—
AIPS_1 (Peripheral Bridge A)	6	Address[31:29] = 110b	0,1,2,3,4,5,6,7
AIPS_0 (Peripheral Bridge B)	7	Address[31:29] = 111b	0,1,2,3,4,5,6,7

1. Master ports 2,3,4,5,6,7 access the flash memory module through slave port 3.

9.1.2 Unimplemented PRS_n and CRS_n registers

On this chip, the PRS5 and CRS5 registers are not implemented. In [Memory Map / Register Definition](#), the locations of these registers are reserved and should not be accessed.

9.1.3 Reset value of PRS_n and CRS_n registers

The reset value of the XBAR_PRS_n registers is 7654_3210h.

The reset value of the XBAR_CRS_n registers is 00FF_0000h.

9.1.4 Master priority elevation

Each implemented XBAR slave has a corresponding [Control Register \(XBAR_CRSn\)](#) whose number is the same as the slave's physical port. Each of these registers has HPE_n fields corresponding to the physical port numbers of the implemented XBAR masters.

In general, set the HPE_n field for a particular master to 1 to allow the elevation of that master's priority of access to the applicable slave. For example, set XBAR_CRs1[HPE6] to 1 to allow the elevation of the SIPI module's priority of access to the EBI.

9.1.4.1 Master priority elevation not used by CSE or FEC

The CSE is connected to XBAR master physical port 6, and the FEC is connected to XBAR master physical port 7. The CSE and FEC cannot signal a request for priority elevation, regardless of the value of the HPE6 and HPE7 fields in every [Control Register \(XBAR_CRSn\)](#).

9.1.4.2 Additional programming for some masters' priority elevation

Enabling the elevation of priority requires additional programming for some masters. The following table summarizes this information. The table refers to fields in the HID1 register that [Hardware Implementation Dependent Register 1 \(HID1\)](#) defines further.

Table 9-3. Additional programming for priority elevation

Master	XBAR physical port	Additional programming	Result
Core0 instruction	0	In addition to setting HPE0 to 1 in XBAR_CRs _n for the applicable slave, set to 1 either or both of these fields in the special-purpose HID1 register for core0: - Elevate External Exceptions (EEE), bit 22 - Elevate Critical Exceptions (ECE), bit 23	Core0's priority is elevated when an event occurs that corresponds to the HID1 field or fields whose value is 1: - External exception when HID1[EEE] is 1 - Critical or NMI exception when HID1[ECE] is 1
Core0 data	1	In addition to setting HPE1 to 1 in XBAR_CRs _n for the applicable slave, set to 1 either or both of these fields in the special-purpose HID1 register for core0: - Elevate External Exceptions (EEE), bit 22 - Elevate Critical Exceptions (ECE), bit 23	Core0's priority is elevated when an event occurs that corresponds to the HID1 field or fields whose value is 1: - External exception when HID1[EEE] is 1 - Critical or NMI exception when HID1[ECE] is 1

Table continues on the next page...

Table 9-3. Additional programming for priority elevation (continued)

Master	XBAR physical port	Additional programming	Result
Core1 instruction	2	In addition to setting HPE2 to 1 in XBAR_CRSn for the applicable slave, set to 1 either or both of these fields in the special-purpose HID1 register for core1: - Elevate External Exceptions (EEE), bit 22 - Elevate Critical Exceptions (ECE), bit 23	Core1's priority is elevated when an event occurs that corresponds to the HID1 field or fields whose value is 1: - External exception when HID1[EEE] is 1 - Critical or NMI exception when HID1[ECE] is 1
Core1 data	3	In addition to setting HPE3 to 1 in XBAR_CRSn for the applicable slave, set to 1 either or both of these fields in the special-purpose HID1 register for core1: - Elevate External Exceptions (EEE), bit 22 - Elevate Critical Exceptions (ECE), bit 23	Core1's priority is elevated when an event occurs that corresponds to the HID1 field or fields whose value is 1: - External exception when HID1[EEE] is 1 - Critical or NMI exception when HID1[ECE] is 1
eDMA_A	4	In addition to setting HPE4 to 1 in XBAR_CRSn for the applicable slave, set eDMA_A's TCD[BWC] field to 01b.	Only eDMA_A can elevate its priority for the applicable slave.
eDMA_B	5	In addition to setting HPE5 to 1 in XBAR_CRSn for the applicable slave, set eDMA_B's TCD[BWC] field to 01b.	Only eDMA_B can elevate its priority for the applicable slave.

Hardware Implementation Dependent Register 1 (HID1)

The following HID1 details supplement and replace information in the *e200z759n3 Core Reference Manual* (document number e200z759n3CRM).

HID1 is the core register (SPR 1009) that controls XBAR priority elevation.

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	Reserved																					E	E	A	Reserved							
W																						E	E	C								
Reset																						E	E	S								
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 9-4. HID1 field descriptions

Field	Name	Description
0-21	Reserved	This field is reserved.
22	EEE	Elevate External Exceptions 0 Disable XBAR priority elevation for normal, external exceptions 1 Enable XBAR priority elevation for normal, external exceptions
23	ECE	Elevate Critical Exceptions 0 Disable XBAR priority elevation for NMI and critical exceptions 1 Enable XBAR priority elevation for NMI and critical exceptions

Table continues on the next page...

Table 9-4. HID1 field descriptions (continued)

Field	Name	Description
24	ATS	Atomic Status (read-only) See the <i>e200z759n3 Core Reference Manual</i> (document number e200z759n3CRM).
25-31	Reserved	This field is reserved.

9.2 Introduction

The information found here provides information on the layout, configuration, and programming of the crossbar switch.

The crossbar switch connects bus masters and bus slaves using a crossbar switch structure. This structure allows all bus masters to access different bus slaves simultaneously, while providing arbitration among the bus masters when they access the same slave. A variety of bus arbitration methods and attributes may be programmed on a slave-by-slave basis.

9.2.1 Features

The crossbar switch includes these features:

- Symmetric crossbar bus switch implementation
 - Allows concurrent accesses from different masters to different slaves
 - Slave arbitration attributes configured on a slave-by-slave basis
- Up to single-clock 64-bit transfer
- Support for burst transfers of up to 16 beats of data
- Low-Power Park mode support
- Dynamic master priority elevation

9.3 Memory Map / Register Definition

Each slave port of the crossbar switch contains configuration registers. Read- and write-transfers require two bus clock cycles. The registers can be read from and written to only in supervisor mode. Additionally, these registers can be read from or written to only by 32-bit accesses.

A bus error response is returned if an unimplemented location is accessed within the crossbar switch.

The CRS_n and PRS_n registers can be programmed to be read-only to prevent changes to their configuration. After being read-only protected, future writes to them will terminate with an error.

NOTE

This section shows the registers for all eight master and slave ports. If a master or slave is not used on this particular device, then unexpected results occur when writing to its registers. See the chip configuration details for the exact master/slave assignments for your device.

All references to the crossbar switch registers are based on the physical port connections.

XBAR memory map

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
0	Priority Registers Slave (XBAR_PRS0)	32	R/W	See section	9.3.1/447
10	Control Register (XBAR_CRS0)	32	R/W	See section	9.3.2/450
100	Priority Registers Slave (XBAR_PRS1)	32	R/W	See section	9.3.1/447
110	Control Register (XBAR_CRS1)	32	R/W	See section	9.3.2/450
200	Priority Registers Slave (XBAR_PRS2)	32	R/W	See section	9.3.1/447
210	Control Register (XBAR_CRS2)	32	R/W	See section	9.3.2/450
300	Priority Registers Slave (XBAR_PRS3)	32	R/W	See section	9.3.1/447
310	Control Register (XBAR_CRS3)	32	R/W	See section	9.3.2/450
400	Priority Registers Slave (XBAR_PRS4)	32	R/W	See section	9.3.1/447
410	Control Register (XBAR_CRS4)	32	R/W	See section	9.3.2/450
500	Priority Registers Slave (XBAR_PRS5)	32	R/W	See section	9.3.1/447
510	Control Register (XBAR_CRS5)	32	R/W	See section	9.3.2/450
600	Priority Registers Slave (XBAR_PRS6)	32	R/W	See section	9.3.1/447
610	Control Register (XBAR_CRS6)	32	R/W	See section	9.3.2/450

Table continues on the next page...

XBAR memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
700	Priority Registers Slave (XBAR_PRS7)	32	R/W	See section	9.3.1/447
710	Control Register (XBAR_CRS7)	32	R/W	See section	9.3.2/450

9.3.1 Priority Registers Slave (XBAR_PRSn)

The priority registers (PRSn) set the priority of each master port on a per slave port basis and reside in each slave port. The priority register can be accessed only with 32-bit accesses. After the CRSn[RO] bit is set, the PRSn register can only be read; attempts to write to it have no effect on PRSn and result in a bus-error response to the master initiating the write.

Two available masters must not be programmed with the same priority level. Attempts to program two or more masters with the same priority level result in a bus-error response and the PRSn is not updated.

NOTE

Valid values for the Mn priority fields depend on which masters are available on the chip. This information can be found in the chip-specific information for the crossbar.

- If the chip contains less than five masters, values 0 to 3 are valid. Writing other values will result in an error.
- If the chip contains five or more masters, valid values are 0 to n-1, where n is the number of masters attached to the Crossbar Switch. Other values will result in an error.

NOTE

See the chip-specific crossbar information for the reset value of this register.

Address: 0h base + 0h offset + (256d × i), where i=0d to 7d

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0				0				0				0			
W																
Reset	0	0*	0*	0*	0	0*	0*	0*	0	0*	0*	0*	0	0*	0*	0*

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0				0				0				0			
W																
Reset	0	0*	0*	0*	0	0*	0*	0*	0	0*	0*	0*	0	0*	0*	0*

* Notes:

- See the chip-specific crossbar information for the reset value of this register.

XBAR_PRSn field descriptions

Field	Description
0 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
1–3 M7	Master 7 Priority. Sets the arbitration priority for this port on the associated slave port. 000 This master has level 1, or highest, priority when accessing the slave port. 001 This master has level 2 priority when accessing the slave port. 010 This master has level 3 priority when accessing the slave port. 011 This master has level 4 priority when accessing the slave port. 100 This master has level 5 priority when accessing the slave port. 101 This master has level 6 priority when accessing the slave port. 110 This master has level 7 priority when accessing the slave port. 111 This master has level 8, or lowest, priority when accessing the slave port.
4 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
5–7 M6	Master 6 Priority. Sets the arbitration priority for this port on the associated slave port. 000 This master has level 1, or highest, priority when accessing the slave port. 001 This master has level 2 priority when accessing the slave port. 010 This master has level 3 priority when accessing the slave port. 011 This master has level 4 priority when accessing the slave port. 100 This master has level 5 priority when accessing the slave port. 101 This master has level 6 priority when accessing the slave port. 110 This master has level 7 priority when accessing the slave port. 111 This master has level 8, or lowest, priority when accessing the slave port.
8 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
9–11 M5	Master 5 Priority. Sets the arbitration priority for this port on the associated slave port. 000 This master has level 1, or highest, priority when accessing the slave port. 001 This master has level 2 priority when accessing the slave port. 010 This master has level 3 priority when accessing the slave port. 011 This master has level 4 priority when accessing the slave port. 100 This master has level 5 priority when accessing the slave port. 101 This master has level 6 priority when accessing the slave port. 110 This master has level 7 priority when accessing the slave port. 111 This master has level 8, or lowest, priority when accessing the slave port.
12 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
13–15 M4	Master 4 Priority. Sets the arbitration priority for this port on the associated slave port. 000 This master has level 1, or highest, priority when accessing the slave port. 001 This master has level 2 priority when accessing the slave port. 010 This master has level 3 priority when accessing the slave port. 011 This master has level 4 priority when accessing the slave port. 100 This master has level 5 priority when accessing the slave port. 101 This master has level 6 priority when accessing the slave port. 110 This master has level 7 priority when accessing the slave port. 111 This master has level 8, or lowest, priority when accessing the slave port.

Table continues on the next page...

XBAR_PRSn field descriptions (continued)

Field	Description
16 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
17–19 M3	Master 3 Priority. Sets the arbitration priority for this port on the associated slave port. 000 This master has level 1, or highest, priority when accessing the slave port. 001 This master has level 2 priority when accessing the slave port. 010 This master has level 3 priority when accessing the slave port. 011 This master has level 4 priority when accessing the slave port. 100 This master has level 5 priority when accessing the slave port. 101 This master has level 6 priority when accessing the slave port. 110 This master has level 7 priority when accessing the slave port. 111 This master has level 8, or lowest, priority when accessing the slave port.
20 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
21–23 M2	Master 2 Priority. Sets the arbitration priority for this port on the associated slave port. 000 This master has level 1, or highest, priority when accessing the slave port. 001 This master has level 2 priority when accessing the slave port. 010 This master has level 3 priority when accessing the slave port. 011 This master has level 4 priority when accessing the slave port. 100 This master has level 5 priority when accessing the slave port. 101 This master has level 6 priority when accessing the slave port. 110 This master has level 7 priority when accessing the slave port. 111 This master has level 8, or lowest, priority when accessing the slave port.
24 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
25–27 M1	Master 1 Priority. Sets the arbitration priority for this port on the associated slave port. 000 This master has level 1, or highest, priority when accessing the slave port. 001 This master has level 2 priority when accessing the slave port. 010 This master has level 3 priority when accessing the slave port. 011 This master has level 4 priority when accessing the slave port. 100 This master has level 5 priority when accessing the slave port. 101 This master has level 6 priority when accessing the slave port. 110 This master has level 7 priority when accessing the slave port. 111 This master has level 8, or lowest, priority when accessing the slave port.
28 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
29–31 M0	Master 0 Priority. Sets the arbitration priority for this port on the associated slave port. 000 This master has level 1, or highest, priority when accessing the slave port. 001 This master has level 2 priority when accessing the slave port. 010 This master has level 3 priority when accessing the slave port. 011 This master has level 4 priority when accessing the slave port. 100 This master has level 5 priority when accessing the slave port. 101 This master has level 6 priority when accessing the slave port. 110 This master has level 7 priority when accessing the slave port. 111 This master has level 8, or lowest, priority when accessing the slave port.

9.3.2 Control Register (XBAR_CRSn)

These registers control several features of each slave port and must be accessed using 32-bit accesses. After CRSn[RO] is set, the PRSn can only be read; attempts to write to it have no effect and result in an error response.

NOTE

See the chip-specific crossbar information for the reset value of this register.

Not all HPE_n fields may be active. See the chip-specific crossbar information for which masters support high priority elevation. Setting a field corresponding to a master that does not support high-priority elevation has no effect.

Address: 0h base + 10h offset + (256d × i), where i=0d to 7d

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	RO	HLP	0						HPE7	HPE6	HPE5	HPE4	HPE3	HPE2	HPE1	HPE0
W																
Reset	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0						ARB		0		PCTL		0	PARK		
W																
Reset	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*

* Notes:

- See the chip-specific crossbar information for the reset value of this register.

XBAR_CRSn field descriptions

Field	Description
0 RO	Read Only Forces the PRSn and CRSn registers to be read-only. After being set, only a hardware reset clears this field. 0 The CRSn and PRSn registers are writeable 1 The CRSn and PRSn registers are read-only and cannot be written (attempted writes have no effect on the registers and result in a bus error response).
1 HLP	Halt Low Priority Sets the initial arbitration priority for low power mode requests. Setting this bit will not affect the request for low power mode from attaining highest priority once it has control of the slave ports. 0 The low power mode request has the highest priority for arbitration on this slave port 1 The low power mode request has the lowest initial priority for arbitration on this slave port

Table continues on the next page...

XBAR_CRSn field descriptions (continued)

Field	Description
2–7 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
8 HPE7	On this slave port, enable priority elevation for master 7. If enabled, the master is able to elevate its priority to the highest. 0 Priority elevation for master 7 is disabled on this slave port 1 Priority elevation for master 7 is enabled on this slave port
9 HPE6	On this slave port, enable priority elevation for master 6. If enabled, the master is able to elevate its priority to the highest. 0 Priority elevation for master 6 is disabled on this slave port 1 Priority elevation for master 6 is enabled on this slave port
10 HPE5	On this slave port, enable priority elevation for master 5. If enabled, the master is able to elevate its priority to the highest. 0 Priority elevation for master 5 is disabled on this slave port 1 Priority elevation for master 5 is enabled on this slave port
11 HPE4	On this slave port, enable priority elevation for master 4. If enabled, the master is able to elevate its priority to the highest. 0 Priority elevation for master 4 is disabled on this slave port 1 Priority elevation for master 4 is enabled on this slave port
12 HPE3	On this slave port, enable priority elevation for master 3. If enabled, the master is able to elevate its priority to the highest. 0 Priority elevation for master 3 is disabled on this slave port 1 Priority elevation for master 3 is enabled on this slave port
13 HPE2	On this slave port, enable priority elevation for master 2. If enabled, the master is able to elevate its priority to the highest. 0 Priority elevation for master 2 is disabled on this slave port 1 Priority elevation for master 2 is enabled on this slave port
14 HPE1	On this slave port, enable priority elevation for master 1. If enabled, the master is able to elevate its priority to the highest. 0 Priority elevation for master 1 is disabled on this slave port 1 Priority elevation for master 1 is enabled on this slave port
15 HPE0	On this slave port, enable priority elevation for master 0. If enabled, the master is able to elevate its priority to the highest. 0 Priority elevation for master 0 is disabled on this slave port 1 Priority elevation for master 0 is enabled on this slave port
16–21 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
22–23 ARB	Arbitration Mode Selects the arbitration policy for the slave port. 00 Fixed priority 01 Round-robin (rotating) priority

Table continues on the next page...

XBAR_CRSn field descriptions (continued)

Field	Description
	10 Reserved 11 Reserved
24–25 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
26–27 PCTL	Parking Control Determines the slave port's parking control. The low-power park feature results in an overall power savings if the slave port is not saturated; however, this forces an extra latency clock when any master tries to access the slave port while not in use because it is not parked on any master. 00 When no master makes a request, the arbiter parks the slave port on the master port defined by the PARK bit field 01 When no master makes a request, the arbiter parks the slave port on the last master to be in control of the slave port 10 Low-power park. When no master makes a request, the slave port is not parked on a master and the arbiter drives all outputs to a constant safe state 11 Reserved
28 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
29–31 PARK	Park Determines which master port the current slave port parks on when no masters are actively making requests and the PCTL bits are cleared. NOTE: Select only master ports that are present on the chip. Otherwise, undefined behavior might occur. 000 Park on master port M0 001 Park on master port M1 010 Park on master port M2 011 Park on master port M3 100 Park on master port M4 101 Park on master port M5 110 Park on master port M6 111 Park on master port M7

9.4 Functional Description

9.4.1 General operation

When a master accesses the crossbar switch, the access is immediately taken. If the targeted slave port of the access is available, then the access is immediately presented on the slave port. Single-clock or zero-wait-state accesses are possible through the crossbar. If the targeted slave port of the access is busy or parked on a different master port, the

requesting master simply sees wait states inserted until the targeted slave port can service the master's request. The latency in servicing the request depends on each master's priority level and the responding slave's access time.

Because the crossbar switch appears to be just another slave to the master device, the master device has no knowledge of whether it actually owns the slave port it is targeting. While the master does not have control of the slave port it is targeting, it simply waits.

After the master has control of the slave port it is targeting, the master remains in control of the slave port until it relinquishes the slave port by running an IDLE cycle or by targeting a different slave port for its next access.

The master can also lose control of the slave port if another higher-priority master makes a request to the slave port. However, if the master is running a fixed-length burst transfer it retains control of the slave port until that transfer completes.

The crossbar terminates all master IDLE transfers, as opposed to allowing the termination to come from one of the slave buses. Additionally, when no master is requesting access to a slave port, the crossbar drives IDLE transfers onto the slave bus, even though a default master may be granted access to the slave port.

When a slave bus is being idled by the crossbar, it can park the slave port on the master port indicated by `CRSn[PARK]`. This is done to save the initial clock of arbitration delay that otherwise would be seen if the same master had to arbitrate to gain control of the slave port. The slave port can also be put into Low Power Park mode to save power, by using `CRSn[PCTL]`.

9.4.2 Register coherency

The operation of the crossbar is affected as soon as a register is written. The values of the registers do not track with slave-port-related master accesses, but instead track only with slave accesses.

9.4.3 Arbitration

The crossbar switch supports two arbitration algorithms:

- Fixed priority
- Round-robin

The arbitration scheme is independently programmable for each slave port.

9.4.3.1 Fixed-priority operation

When operating in fixed-priority mode, each master is assigned a unique priority level in the priority registers (PRSn). If two masters request access to the same slave port, the master with the highest priority in the selected priority register gains control over the slave port.

NOTE

In this arbitration mode, a higher-priority master can monopolize a slave port, preventing accesses from any lower-priority master to the port.

When a master makes a request to a slave port, the slave port checks whether the new requesting master's priority level is higher than that of the master that currently has control over the slave port, unless the slave port is in a parked state. The slave port performs an arbitration check at every clock edge to ensure that the proper master, if any, has control of the slave port.

The following table describes possible scenarios based on the requesting master port:

Table 9-24. How the Crossbar Switch grants control of a slave port to a master

When	Then the Crossbar Switch grants control to the requesting master
Both of the following are true: - The current master is not running a transfer. - The new requesting master's priority level is higher than that of the current master.	At the next clock edge
Both of the following are true: - The current master is running a fixed length burst transfer or a locked transfer. - The requesting master's priority level is higher than that of the current master.	At the end of the burst transfer or locked transfer
The requesting master's priority level is lower than the current master.	At the conclusion of one of the following cycles: - An IDLE cycle - A non-IDLE cycle to a location other than the current slave port

9.4.3.2 Round-robin priority operation

When operating in round-robin mode, each master is assigned a relative priority based on the master port number. This relative priority is compared to the master port number (ID) of the last master to perform a transfer on the slave bus. The highest priority requesting

master becomes owner of the slave bus at the next transfer boundary, accounting for locked and fixed-length burst transfers. Priority is based on how far ahead the ID of the requesting master is to the ID of the last master.

After granted access to a slave port, a master may perform as many transfers as desired to that port until another master makes a request to the same slave port. The next master in line is granted access to the slave port at the next transfer boundary, or possibly on the next clock cycle if the current master has no pending access request.

As an example of arbitration in round-robin mode, assume the crossbar is implemented with master ports 0, 1, 4, and 5. If the last master of the slave port was master 1, and master 0, 4, and 5 make simultaneous requests, they are serviced in the order: 4 then 5 then 0.

Parking may continue to be used in a round-robin mode, but does not affect the round-robin pointer unless the parked master actually performs a transfer. Handoff occurs to the next master in line after one cycle of arbitration. If the slave port is put into low-power park mode, the round-robin pointer is reset to point at master port 0, giving it the highest priority.

9.4.3.3 Priority assignment

Each master port must be assigned a unique 3-bit priority level. If an attempt is made to program multiple master ports with the same priority level within the priority registers (PRSn), the crossbar switch responds with a bus error and the registers are not updated.

9.5 Initialization/application information

No initialization is required for the crossbar switch.

Hardware reset ensures that all the register bits used by the crossbar switch are properly initialized to a valid state. However, settings and priorities may be programmed to achieve maximum system performance.

NOTE

- During configuration of the crossbar switch, all other masters must be idle.
- To prevent reconfiguration of the crossbar switch, write 1 to CRSn[RO].

Chapter 10

Crossbar Integrity Checker (XBIC)

10.1 Chip-specific XBIC information

This chip has one instance of the XBIC module.

10.1.1 XBIC master and slave assignments

The XBIC identifies each XBAR master and slave in terms of the master or slave's physical port number. See the "Physical master port" assignments in [Table 9-1](#) and the "Slave port" assignments in [Table 9-2](#).

10.1.2 Unimplemented MCR and ESR fields

On this chip, the MCR[SE5] and ESR[DPSE5] fields are not implemented. In [XBIC Module Control Register \(XBIC_MCR\)](#) and [XBIC Error Status Register \(XBIC_ESR\)](#), these fields are reserved.

10.2 Overview

The Crossbar Integrity Checker (XBIC) verifies the integrity of the crossbar transfers.

10.3 Features

The XBIC has the following features:

- Verification of attribute information for all crossbar transfers

- EDC (72,64) code protects against single and double bit errors
- Verification of feedback information for each data phase during crossbar transfer
- Error injection for testing
 - Programmable master and slave port specifiers
 - Programmable 8-bit toggle vector to insert error in master EDC checkbit value
 - Address, EDC syndrome, master and slave port information captured on error
- Programmable integrity check enable on a per-slave-port basis
- Programmable integrity feedback check enable on a per-master-port basis

10.4 Block diagram

The crossbar transfer attribute information for all master and slave ports is routed to the XBIC, which calculates and checks the EDC parity over the attribute information as shown in the following diagram.

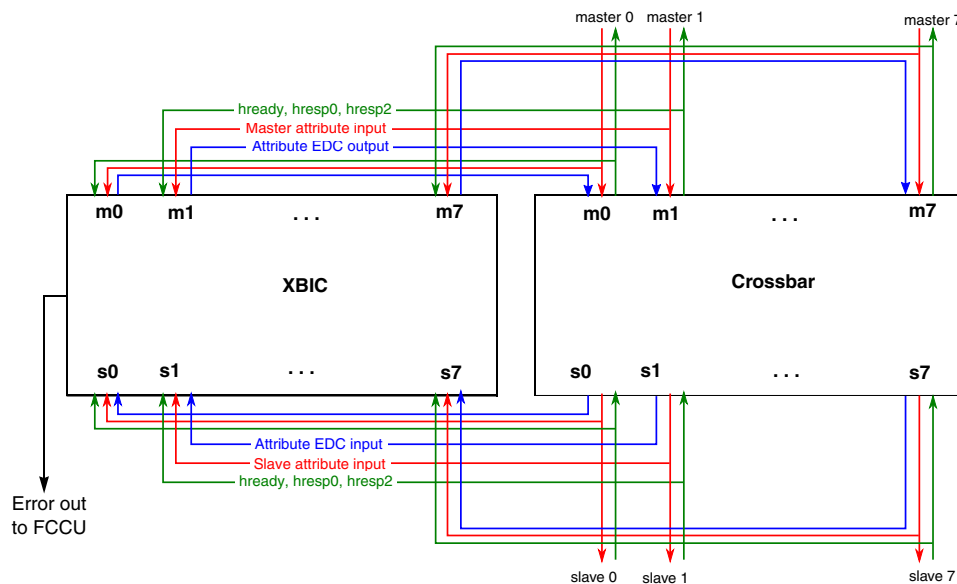


Figure 10-1. XBIC system block diagram

10.5 External signal description

The XBIC has no external interface signals.

10.6 Memory map and register definition

The XBIC programming model has four 32-bit registers. The programming model can only be accessed in supervisor mode using 32-bit (word) accesses. Attempted references with a different access size, to an undefined (reserved) address, with a non-supported access type (a write to a read-only register), or in user mode generate an error termination.

XBIC memory map

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
0	XBIC Module Control Register (XBIC_MCR)	32	R/W	FFFF_0000h	10.6.1/459
4	XBIC Error Injection Register (XBIC_EIR)	32	R/W	0000_0000h	10.6.2/461
8	XBIC Error Status Register (XBIC_ESR)	32	R	0000_0000h	10.6.3/462
C	XBIC Error Address Register (XBIC_EAR)	32	R	0000_0000h	10.6.4/464

10.6.1 XBIC Module Control Register (XBIC_MCR)

The XBIC_MCR allows attribute integrity checking to be enabled on a per-port basis.

Address: 0h base + 0h offset = 0h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	SE0	SE1	SE2	SE3	SE4	SE5	SE6	SE7	ME0	ME1	ME2	ME3	ME4	ME5	ME6	ME7
W																
Reset	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

XBIC_MCR field descriptions

Field	Description
0 SE0	Slave Port Enable for EDC error detect. 0 Attribute integrity checking disabled for slave port 0. 1 Attribute integrity checking enabled for slave port 0.
1 SE1	Slave Port Enable for EDC error detect. 0 Attribute integrity checking disabled for slave port 1. 1 Attribute integrity checking enabled for slave port 1.

Table continues on the next page...

XBIC_MCR field descriptions (continued)

Field	Description
2 SE2	Slave Port Enable for EDC error detect. 0 Attribute integrity checking disabled for slave port 2. 1 Attribute integrity checking enabled for slave port 2.
3 SE3	Slave Port Enable for EDC error detect. 0 Attribute integrity checking disabled for slave port 3. 1 Attribute integrity checking enabled for slave port 3.
4 SE4	Slave Port Enable for EDC error detect. 0 Attribute integrity checking disabled for slave port 4. 1 Attribute integrity checking enabled for slave port 4.
5 SE5	Slave Port Enable for EDC error detect. 0 Attribute integrity checking disabled for slave port 5. 1 Attribute integrity checking enabled for slave port 5.
6 SE6	Slave Port Enable for EDC error detect. 0 Attribute integrity checking disabled for slave port 6. 1 Attribute integrity checking enabled for slave port 6.
7 SE7	Slave Port Enable for EDC error detect. 0 Attribute integrity checking disabled for slave port 7. 1 Attribute integrity checking enabled for slave port 7.
8 ME0	Master Port Enable for slave driven signal safety check. 0 Attribute integrity checking disabled for master port 0. 1 Attribute integrity checking enabled for master port 0.
9 ME1	Master Port Enable for slave driven signal safety check. 0 Attribute integrity checking disabled for master port 1. 1 Attribute integrity checking enabled for master port 1.
10 ME2	Master Port Enable for slave driven signal safety check. 0 Attribute integrity checking disabled for master port 2. 1 Attribute integrity checking enabled for master port 2.
11 ME3	Master Port Enable for slave driven signal safety check. 0 Attribute integrity checking disabled for master port 3. 1 Attribute integrity checking enabled for master port 3.
12 ME4	Master Port Enable for slave driven signal safety check. 0 Attribute integrity checking disabled for master port 4. 1 Attribute integrity checking enabled for master port 4.
13 ME5	Master Port Enable for slave driven signal safety check. 0 Attribute integrity checking disabled for master port 5. 1 Attribute integrity checking enabled for master port 5.

Table continues on the next page...

XBIC_MCR field descriptions (continued)

Field	Description
14 ME6	Master Port Enable for slave driven signal safety check. 0 Attribute integrity checking disabled for master port 6. 1 Attribute integrity checking enabled for master port 6.
15 ME7	Master Port Enable for slave driven signal safety check. 0 Attribute integrity checking disabled for master port 7. 1 Attribute integrity checking enabled for master port 7.
16–31 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

10.6.2 XBIC Error Injection Register (XBIC_EIR)

The XBIC_EIR contains fields for controlling the XBIC error injection function. When an error is injected, the EDC syndrome associated with the programmed master and slave ports is modified, causing the XBIC to assert the error indication to the FCCU and capturing the transfer information in the XBIC_ESR and XBIC_EAR registers. Otherwise, transfers are not affected by XBIC error injection.

Address: 0h base + 4h offset = 4h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	EIE	0														
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	SLV			MST				SYN							
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

XBIC_EIR field descriptions

Field	Description
0 EIE	Error Injection Enable. 0 Error injection disabled 1 Error injection enabled
1–16 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
17–19 SLV	Target Slave Port. Error injection is enabled for the designated slave port. Other slave ports are unaffected.
20–23 MST	Target Master ID. Error injection is enabled for the designated master ID. Transfers with other master IDs are not affected.
24–31 SYN	Syndrome. This value is exclusive-ored with the calculated syndrome (which should be zero) to generate an error with the specified syndrome. A value of zero does not generate an error.

10.6.3 XBIC Error Status Register (XBIC_ESR)

The Error Status Register (XBIC_ESR) contains two types of error information about the last transfer. If an attribute integrity check error was detected, the slave port, the master port, and the syndrome are captured in the SLV, MST, and SYN fields. If there is a mismatch among internal signals (hready, hresp0, and hresp2) during the data phase, the slave and master port are catured in the DPSE0-7 and DPME0-7 fields.

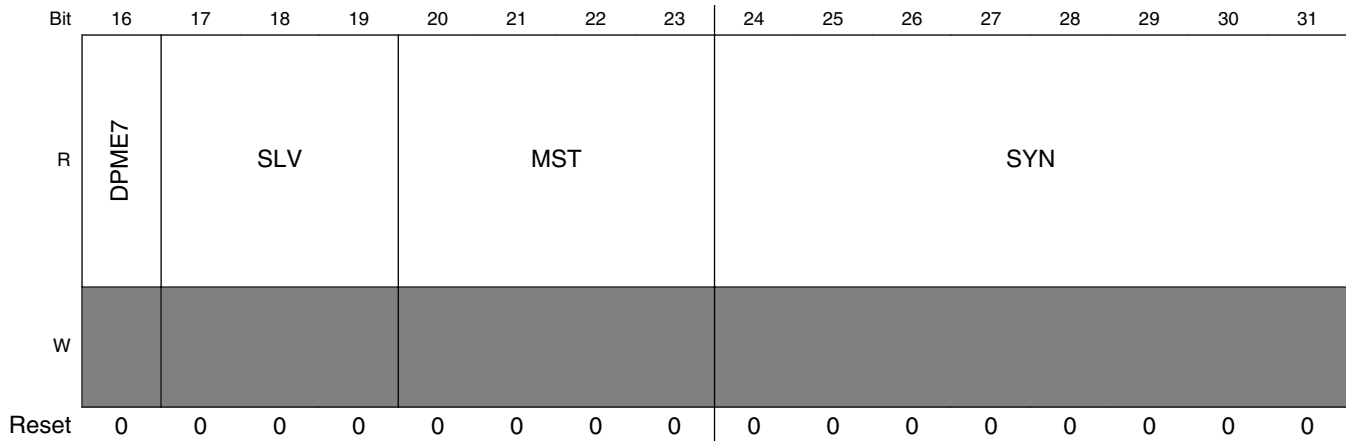
This register is cleared only on reset.

NOTE

When the XBIC detects back-to-back faults on a system bus path through the crossbar switch, the fault information captured in the XBIC_ESR does not correspond to the initial fault event but rather the subsequent fault event. While the fault event is properly detected, diagnostic status information in the XBIC_ESR register describing the initial fault event is lost. This can only occur in the event of a series of bus transactions targeting the same crossbar slave target where the series of bus transactions are not separated by idle or stall cycles.

Address: 0h base + 8h offset = 8h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	VLD	DPSE0	DPSE1	DPSE2	DPSE3	DPSE4	DPSE5	DPSE6	DPSE7	DPME0	DPME1	DPME2	DPME3	DPME4	DPME5	DPME6
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

**XBIC_ESR field descriptions**

Field	Description
0 VLD	Error Status Valid. 0 No error detected, other fields of the ESR and EAR are not valid. 1 Error detected, all fields of the ESR and EAR are valid.
1 DPSE0	Data phase slave port error. 0 No error on slave port 0 1 Data phase error on slave port 0
2 DPSE1	Data phase slave port error. 0 No error on slave port 1 1 Data phase error on slave port 1
3 DPSE2	Data phase slave port error. 0 No error on slave port 2 1 Data phase error on slave port 2
4 DPSE3	Data phase slave port error. 0 No error on slave port 3 1 Data phase error on slave port 3
5 DPSE4	Data phase slave port error. 0 No error on slave port 4 1 Data phase error on slave port 4
6 DPSE5	Data phase slave port error. 0 No error on slave port 5 1 Data phase error on slave port 5
7 DPSE6	Data phase slave port error. 0 No error on slave port 6 1 Data phase error on slave port 6
8 DPSE7	Data phase slave port error.

Table continues on the next page...

XBIC_ESR field descriptions (continued)

Field	Description
	0 No error on slave port 7 1 Data phase error on slave port 7
9 DPME0	Data phase master port error. 0 No error on master port 0 1 Data phase error on master port 0
10 DPME1	Data phase master port error. 0 No error on master port 1 1 Data phase error on master port 1
11 DPME2	Data phase master port error. 0 No error on master port 2 1 Data phase error on master port 2
12 DPME3	Data phase master port error. 0 No error on master port 3 1 Data phase error on master port 3
13 DPME4	Data phase master port error. 0 No error on master port 4 1 Data phase error on master port 4
14 DPME5	Data phase master port error. 0 No error on master port 5 1 Data phase error on master port 5
15 DPME6	Data phase master port error. 0 No error on master port 6 1 Data phase error on master port 6
16 DPME7	Data phase master port error. 0 No error on master port 7 1 Data phase error on master port 7
17–19 SLV	Slave Port. The slave port targeted by the last transfer with an error.
20–23 MST	Master ID. The master ID value of the last transfer with an error.
24–31 SYN	Syndrom. The syndrome calculated for the last transfer with an error. For single bit errors the signal in error can be determined, see Table 10-6 .

10.6.4 XBIC Error Address Register (XBIC_EAR)

The Error Address Register (XBIC_EAR) contains the address of the first transfer in which an attribute integrity check error was detected on an enabled slave port. This register is only cleared on reset.

NOTE

When the XBIC detects back-to-back faults on a system bus path through the crossbar switch, the fault information captured in the XBIC_EAR does not correspond to the initial fault event but rather the subsequent fault event. While the fault event is properly detected, diagnostic status information in the XBIC_ESR register describing the initial fault event is lost. This can only occur in the event of a series of bus transactions targeting the same crossbar slave target where the series of bus transactions are not separated by idle or stall cycles.

Address: 0h base + Ch offset = Ch

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	ADDR																															
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	

XBIC_EAR field descriptions

Field	Description
0–31 ADDR	The address of the last transfer with an error.

10.7 Functional description

The Crossbar Integrity Checker (XBIC) verifies the integrity of the crossbar interface. The Module Control Register (XBIC_MCR) allows integrity checking to be enabled on a individual port basis. When an error is detected, it is captured in XBIC_ESR register and XBIC also sends out an error signal to the Fault Collection and Control Unit (FCCU). Detection of an error does not generate a bus error. The XBIC integrity checking is independent from the end-to-end ECC that covers the transfer address and data.

During the AHB address phase, the crossbar attribute information is checked using an 8-bit Error Detection Code (EDC), which detects any single- or double-bit errors. When an error is detected, due either to hardware fault or error injection, information related to the error is captured and reported in XBIC_ESR and XBIC_EAR. The cause of a single-bit error can be determined by the syndrome value reported in the XBIC_ESR as shown in [Table 10-6](#). Any other syndrome indicates a multi-bit error.

During the AHB data phase, the responding signals from the slave to master are checked when passing through the crossbar. When an error is detected, the information is captured in the XBIC_ESR DPSE0-7 and DPME0-7 fields.

The XBIC can also be programmed to intentionally inject EDC errors for testing. Error injection is targeted toward a single slave port and a single master ID at a time as determined by the settings in the Error Injection Register (XBIC_EIR). When an error is injected, the EDC syndrome is modified which causes the XBIC to assert the error indication to the FCCU. Otherwise transfers are unaffected by XBIC error injection. This approach allows the check logic to be verified without compromising the integrity of the data transfer. Once the error injection function is enabled by setting XBIC_EIR[EIE], errors are induced on all subsequent targeted transactions until XBIC_EIR[EIE] is cleared

Table 10-6. Hexadecimal attribute single-bit error syndromes

Signal	SYN	Signal	SYN	Signal	SYN	Signal	SYN
hwrite	07	hbstrb[7]	70	hdecor[31]	25	hdecor[15]	23
htrans[0]	0b	hbstrb[6]	32	hdecor[30]	68	hdecor[14]	51
hsize[2]	0d	hbstrb[5]	52	hdecor[29]	c7	hdecor[13]	54
hsize[1]	0e	hbstrb[4]	a8	hdecor[28]	83	hdecor[12]	61
hsize[0]	13	hbstrb[3]	43	hdecor[27]	85	hdecor[11]	e3
hprot[5]	15	hbstrb[2]	45	hdecor[26]	86	hdecor[10]	e6
hprot[4]	16	hbstrb[1]	4c	hdecor[25]	89	hdecor[9]	f8
hprot[3]	19	hbstrb[0]	a4	hdecor[24]	8a	hdecor[8]	38
hprot[2]	1a	hmaster[3]	a2	hdecor[23]	8c	hdecor[7]	58
hprot[1]	1c	hmaster[2]	b0	hdecor[22]	49	hdecor[6]	37
hprot[0]	91	hmaster[1]	c1	hdecor[21]	92	hdecor[5]	f1
hburst[2]	a1	hmaster[0]	c2	hdecor[20]	94	hdecor[4]	3b
hburst[1]	64	hslave[2]	c4	hdecor[19]	98	hdecor[3]	3d
hburst[0]	29	hslave[1]	c8	hdecor[18]	46	hdecor[2]	3e
hmastlock	2a	hslave[0]	d0	hdecor[17]	34	hdecor[1]	4f
hunalign	2c	hdecorated	e0	hdecor[16]	4a	hdecor[0]	6e
edc[7]	80	edc[6]	40	edc[5]	20	edc[4]	10
edc[3]	08	edc[2]	04	edc[1]	02	edc[0]	01

Chapter 11

Peripheral Bridge (AIPS/PBRIDGE)

11.1 Chip-specific AIPS/PBRIDGE information

This chip has two instances of the AIPS (PBRIDGE) module:

- AIPS_0 (PBRIDGE_B)
- AIPS_1 (PBRIDGE_A)

11.1.1 Modules assigned to PACR n and OPACR n fields

Each module attached to the Peripheral Bridges is assigned to a memory map slot corresponding to an AIPS register field.

- Every on-platform module has an assigned PACR n field within the PACRA to PACRD registers.
- Every off-platform module has an assigned OPACR n field within the OPACRA to OPACRH registers.

The [Peripheral Access Control Register \(AIPS_PACR \$n\$ \)](#) description provides the location of each PACR n field, and the [Off-Platform Peripheral Access Control Register \(AIPS_OPACR \$n\$ \)](#) description provides the location of each OPACR n field.

The following tables provide the module slot assignments for this chip.

NOTE

Unused PACR n and OPACR n fields are reserved.

Table 11-1. AIPS_0 (PBRIDGE_B) PACR_n field assignments

Module	PACR field
AIPS_0	0
XBAR	1
XBIC	2
MPU_0	3
MPU_1	4
CSE	5
TDM	6
PRAMC_0	7
PRAMC_1	8
SEMA4	9
SWT_B	13
SWT_A	14
STM_A	15
eDMA_A	17
INTC	18
FEC	19
PCU	20
eDMA_B	21
ERM	24
EIM	25
PFLASH	26
PCM	27

Table 11-2. AIPS_0 (PBRIDGE_B) OPACR_n field assignments

Module	OPACR
EQADC_A	0
DECFILTER_A to DECFILTER_H	2
DECFILTER_I to DECFILTER_L	3
DSPI_A	4
DSPI_B	5
DSPI_C	6
ESCI_A	12
ESCI_B	13
ESCI_C	14
FlexCAN_A	16
FlexCAN_B	17

Table continues on the next page...

**Table 11-2. AIPS_0 (PBRIDGE_B) OPACR_n field assignments
(continued)**

Module	OPACR
LFAST_0	22
C55FMC	26
PASS	29
SSCM	30
BAM	31
ETPU_C	32
REACM2	33
ETPU_C PARAM	34
ETPU_C PARAM	35
ETPU_C CODE	36
GLITCH_FILTER_0	40
M_CAN_0	44
M_CAN RAM	45
M_CAN_1	46
PSI5_0	47
SDADC_2	49
SDADC_4	50
SRX_0	55
EMIOS_0	57
CRC	58
SIPI_0	59
PCS	60
PLLDIG	61
DTS	62
JDC	63

Table 11-3. AIPS_1 (PBRIDGE_A) PACR_n field assignments

Module	PACR
AIPS_1	0

Table 11-4. AIPS_1 (PBRIDGE_A) OPACR_n field assignments

Peripheral	OPACR
FCCU	0
EBI configuration	1
SIU_A	4

Table continues on the next page...

**Table 11-4. AIPS_1 (PBRIDGE_A) OPACR_n field assignments
(continued)**

Peripheral	OPACR
SIU_B	6
EMIOS_1	8
ESCI_D	12
ESCI_E	13
ESCI_F	14
PMU	15
ETPU_A/B	16
ETPU_A/B PARAM	18
ETPU_A/B PARAM	19
ETPU_A/B CODE	20
ETPU_A/B CODE	21
DSPI_D	23
DSPI_E	24
PIT_RTI	28
STCU2	29
GLITCH_FILTER_1	33
PSI5_1	37
SDADC_1	49
SDADC_3	50
EQADC_B	53
SRX_1	55
FlexCAN_C	56
FlexCAN_D	57
CMU_0 to CMU_8	59

11.1.2 AIPS/PBRIDGE register reset values

The following table identifies chip-specific reset values of AIPS/PBRIDGE registers.

Table 11-5. AIPS/PBRIDGE register reset values

Register	AIPS_0 (PBRIDGE_B)	AIPS_1 (PBRIDGE_A)
MPRA	7777_7777h	7777_7777h
MPRB	7777_7777h	7777_7777h
PACRA	5444_4444h	5000_0000h
PACRB	4400_0444h	0000_0000h
PACRC	0444_4400h	0000_0000h

Table continues on the next page...

Table 11-5. AIPS/PBRIDGE register reset values (continued)

Register	AIPS_0 (PBRIDGE_B)	AIPS_1 (PBRIDGE_A)
PACRD	4444_0000h	0000_0000h
OPACRA, OPACRB, and OPACRC	4444_4444h	4444_4444h
OPACRD	4454_4444h	4444_4444h
OPACRE, OPACRF, OPACRG, and OPACRH	4444_4444h	4444_4444h

11.2 Introduction

The peripheral bridge converts the crossbar switch interface to an interface that can access most of the slave peripherals on this chip.

The peripheral bridge occupies 64 MB of the address space, which is divided into peripheral slots of 16 KB. (It might be possible that all the peripheral slots are not used. See the memory map chapter for details on slot assignments.) The bridge includes separate clock enable inputs for each of the slots to accommodate slower peripherals.

11.2.1 Features

Key features of the peripheral bridge are:

- Supports peripheral slots with 8-, 16-, and 32-bit datapath width
- Supports a pair of 32-bit transactions for selected 64-bit memory accesses
- Programming model provides memory protection functionality

11.2.2 General operation

The slave devices connected to the peripheral bridge are modules which contain a programming model of control and status registers. The system masters read and write these registers through the peripheral bridge.

The register maps of the peripherals are located on 16 KB boundaries. Each peripheral is allocated one or more 16-KB block(s) of the memory map.

11.3 Memory map/register definition

The 32-bit peripheral bridge registers can be accessed only in supervisor mode by trusted bus masters. Additionally, these registers must be read from or written to only by a 32-bit aligned access. The peripheral bridge registers are mapped into the Peripheral Access Control Register A PACRA[PACR0] address space.

AIPS memory map

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
0	Master Privilege Register A (AIPS_MPRA)	32	R/W	See section	11.3.1/472
4	Master Privilege Register B (AIPS_MPRB)	32	R/W	See section	11.3.2/476
20	Peripheral Access Control Register (AIPS_PACRA)	32	R/W	See section	11.3.3/478
24	Peripheral Access Control Register (AIPS_PACRB)	32	R/W	See section	11.3.3/478
28	Peripheral Access Control Register (AIPS_PACRC)	32	R/W	See section	11.3.3/478
2C	Peripheral Access Control Register (AIPS_PACRD)	32	R/W	See section	11.3.3/478
40	Off-Platform Peripheral Access Control Register (AIPS_OPACRA)	32	R/W	See section	11.3.4/483
44	Off-Platform Peripheral Access Control Register (AIPS_OPACRB)	32	R/W	See section	11.3.4/483
48	Off-Platform Peripheral Access Control Register (AIPS_OPACRC)	32	R/W	See section	11.3.4/483
4C	Off-Platform Peripheral Access Control Register (AIPS_OPACRD)	32	R/W	See section	11.3.4/483
50	Off-Platform Peripheral Access Control Register (AIPS_OPACRE)	32	R/W	See section	11.3.4/483
54	Off-Platform Peripheral Access Control Register (AIPS_OPACRF)	32	R/W	See section	11.3.4/483
58	Off-Platform Peripheral Access Control Register (AIPS_OPACRG)	32	R/W	See section	11.3.4/483
5C	Off-Platform Peripheral Access Control Register (AIPS_OPACRH)	32	R/W	See section	11.3.4/483

11.3.1 Master Privilege Register A (AIPS_MPRA)

The MPRA specifies identical 4-bit fields defining the access-privilege level associated with a bus master to various peripherals on the chip. The register provides one field per bus master.

NOTE

At reset, the default value loaded into the MPRA fields is chip-specific. See the chip configuration details for the value of a particular device.

A register field that maps to an unimplemented master or peripheral behaves as read-only-zero.

Each master is assigned a logical ID from 0 to 15. See the master logical ID assignment table in the chip-specific AIPS information.

Address: 0h base + 0h offset = 0h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	MTR0	MTW0	MPL0	0	MTR1	MTW1	MPL1	0	MTR2	MTW2	MPL2	0	MTR3	MTW3	MPL3
W																
Reset	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	MTR4	MTW4	MPL4	0	MTR5	MTW5	MPL5	0	MTR6	MTW6	MPL6	Reserved			
W																
Reset	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*			

* Notes:

- The reset value is chip-dependent and can be found in the chip-specific AIPS information.

AIPS_MPRA field descriptions

Field	Description
0 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
1 MTR0	Master 0 Trusted For Read Determines whether the master is trusted for read accesses. 0 This master is not trusted for read accesses. 1 This master is trusted for read accesses.
2 MTW0	Master 0 Trusted For Writes Determines whether the master is trusted for write accesses. 0 This master is not trusted for write accesses. 1 This master is trusted for write accesses.
3 MPL0	Master 0 Privilege Level Specifies how the privilege level of the master is determined. 0 Accesses from this master are forced to user-mode. 1 Accesses from this master are not forced to user-mode.
4 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

Table continues on the next page...

AIPS_MPRA field descriptions (continued)

Field	Description
5 MTR1	Master 1 Trusted for Read Determines whether the master is trusted for read accesses. 0 This master is not trusted for read accesses. 1 This master is trusted for read accesses.
6 MTW1	Master 1 Trusted for Writes Determines whether the master is trusted for write accesses. 0 This master is not trusted for write accesses. 1 This master is trusted for write accesses.
7 MPL1	Master 1 Privilege Level Specifies how the privilege level of the master is determined. 0 Accesses from this master are forced to user-mode. 1 Accesses from this master are not forced to user-mode.
8 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
9 MTR2	Master 2 Trusted For Read Determines whether the master is trusted for read accesses. 0 This master is not trusted for read accesses. 1 This master is trusted for read accesses.
10 MTW2	Master 2 Trusted For Writes Determines whether the master is trusted for write accesses. 0 This master is not trusted for write accesses. 1 This master is trusted for write accesses.
11 MPL2	Master 2 Privilege Level Specifies how the privilege level of the master is determined. 0 Accesses from this master are forced to user-mode. 1 Accesses from this master are not forced to user-mode.
12 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
13 MTR3	Master 3 Trusted For Read Determines whether the master is trusted for read accesses. 0 This master is not trusted for read accesses. 1 This master is trusted for read accesses.
14 MTW3	Master 3 Trusted For Writes Determines whether the master is trusted for write accesses. 0 This master is not trusted for write accesses. 1 This master is trusted for write accesses.

Table continues on the next page...

AIPS_MPRA field descriptions (continued)

Field	Description
15 MPL3	Master 3 Privilege Level Specifies how the privilege level of the master is determined. 0 Accesses from this master are forced to user-mode. 1 Accesses from this master are not forced to user-mode.
16 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
17 MTR4	Master 4 Trusted For Read Determines whether the master is trusted for read accesses. 0 This master is not trusted for read accesses. 1 This master is trusted for read accesses.
18 MTW4	Master 4 Trusted For Writes Determines whether the master is trusted for write accesses. 0 This master is not trusted for write accesses. 1 This master is trusted for write accesses.
19 MPL4	Master 4 Privilege Level Specifies how the privilege level of the master is determined. 0 Accesses from this master are forced to user-mode. 1 Accesses from this master are not forced to user-mode.
20 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
21 MTR5	Master 5 Trusted For Read Determines whether the master is trusted for read accesses. 0 This master is not trusted for read accesses. 1 This master is trusted for read accesses.
22 MTW5	Master 5 Trusted For Writes Determines whether the master is trusted for write accesses. 0 This master is not trusted for write accesses. 1 This master is trusted for write accesses.
23 MPL5	Master 5 Privilege Level Specifies how the privilege level of the master is determined. 0 Accesses from this master are forced to user-mode. 1 Accesses from this master are not forced to user-mode.
24 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
25 MTR6	Master 6 Trusted for Read Determines whether the master is trusted for read accesses.

Table continues on the next page...

AIPS_MPRB field descriptions (continued)

Field	Description
	0 This master is not trusted for read accesses. 1 This master is trusted for read accesses.
26 MTW6	Master 6 Trusted for Writes Determines whether the master is trusted for write accesses. 0 This master is not trusted for write accesses. 1 This master is trusted for write accesses.
27 MPL6	Master 6 Privilege Level Specifies how the privilege level of the master is determined. 0 Accesses from this master are forced to user-mode. 1 Accesses from this master are not forced to user-mode.
28–31 Reserved	This field is reserved.

11.3.2 Master Privilege Register B (AIPS_MPRB)

The MPRB registers specifies identical 4-bit fields defining the access-privilege level associated with a bus master to the various peripherals on the chip. The register provides one field per bus master.

NOTE

At reset, the default value loaded into the MPRB fields is chip-specific. See the chip configuration information for the value on your particular device.

Accesses to registers or register fields which correspond to master or peripheral locations which are not implemented return zeros on reads, and are ignored on writes.

Each master is assigned a logical ID from 0 to 15. See the master logical ID assignment table in the AIPS chip-specific information.

Address: 0h base + 4h offset = 4h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0				0				0				0			
W		MTR8	MTW8	MPL8		MTR9	MTW9	MPL9								
Reset	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0				0				0				0			
W																
Reset	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*

* Notes:

- The reset value is chip-dependent and can be found in the chip-specific AIPS information.

AIPS_MPRB field descriptions

Field	Description
0 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
1 MTR8	Master 8 Trusted for Read Determines whether the master is trusted for read accesses. 0 This master is not trusted for read accesses. 1 This master is trusted for read accesses.
2 MTW8	Master 8 Trusted for Writes Determines whether the master is trusted for write accesses. 0 This master is not trusted for write accesses. 1 This master is trusted for write accesses.
3 MPL8	Master 8 Privilege Level Specifies how the privilege level of the master is determined. 0 Accesses from this master are forced to user-mode. 1 Accesses from this master are not forced to user-mode.
4 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
5 MTR9	Master 9 Trusted for Read Determines whether the master is trusted for read accesses. 0 This master is not trusted for read accesses. 1 This master is trusted for read accesses.
6 MTW9	Master 9 Trusted for Writes Determines whether the master is trusted for write accesses. 0 This master is not trusted for write accesses. 1 This master is trusted for write accesses.
7 MPL9	Master 9 Privilege Level Specifies how the privilege level of the master is determined. 0 Accesses from this master are forced to user-mode. 1 Accesses from this master are not forced to user-mode.

Table continues on the next page...

AIPS_MPRB field descriptions (continued)

Field	Description
8–11 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
12–15 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
16–19 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
20–23 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
24–27 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
28–31 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

11.3.3 Peripheral Access Control Register (AIPS_PACRn)

Each PACR register consists of eight 4-bit PACR fields. Each PACR field defines the access levels for a particular on-platform peripheral. The peripheral assignment to each PACR field is defined by the memory map slot of the peripheral. See the chip-specific AIPS information for the field assignment of a particular peripheral.

Every PACR field to which no peripheral is assigned is reserved. Reads to reserved locations return zeros, and writes are ignored.

The following table shows the location of each peripheral slot's PACR field in the PACR registers.

Offset	Register	[0:3]	[4:7]	[8:11]	[12:15]	[16:19]	[20:23]	[24:27]	[28:31]
0x20	PACRA	PACR0	PACR1	PACR2	PACR3	PACR4	PACR5	PACR6	PACR7
0x24	PACRB	PACR8	PACR9	PACR10	PACR11	PACR12	PACR13	PACR14	PACR15
0x28	PACRC	PACR16	PACR17	PACR18	PACR19	PACR20	PACR21	PACR22	PACR23
0x2C	PACRD	PACR24	PACR25	PACR26	PACR27	PACR28	PACR29	PACR30	PACR31

Address: 0h base + 20h offset + (4d × i), where i=0d to 3d

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	SP0	WP0	TP0	0	SP1	WP1	TP1	0	SP2	WP2	TP2	0	SP3	WP3	TP3
W																
Reset	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	SP4	WP4	TP4	0	SP5	WP5	TP5	0	SP6	WP6	TP6	0	SP7	WP7	TP7
W																
Reset	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*

* Notes:

- The reset value is chip-dependent and can be found in the AIPS chip-specific information.

AIPS_PACR_n field descriptions

Field	Description
0 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
1 SP0	Supervisor Protect Determines whether the peripheral requires supervisor privilege level for accesses. When this field is set, the master privilege level must indicate the supervisor access attribute, and the MPRx[MPL _n] control field for the master must be set. If not, access terminates with an error response and no peripheral access initiates. 0 This peripheral does not require supervisor privilege level for accesses. 1 This peripheral requires supervisor privilege level for accesses.
2 WP0	Write Protect Determines whether the peripheral allows write access. When this bit is set and a write access is attempted, access terminates with an error response and no peripheral access initiates. 0 This peripheral allows write accesses. 1 This peripheral is write protected.
3 TP0	Trusted Protect Determines whether the peripheral allows accesses from an untrusted master. When this field is set and an access is attempted by an untrusted master, the access terminates with an error response and no peripheral access initiates. 0 Accesses from an untrusted master are allowed. 1 Accesses from an untrusted master are not allowed.
4 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
5 SP1	Supervisor Protect Determines whether the peripheral requires supervisor privilege level for accesses. When this field is set, the master privilege level must indicate the supervisor access attribute, and the MPRx[MPL _n] control field for the master must be set. If not, access terminates with an error response and no peripheral access initiates. 0 This peripheral does not require supervisor privilege level for accesses. 1 This peripheral requires supervisor privilege level for accesses.
6 WP1	Write Protect Determines whether the peripheral allows write accesses. When this field is set and a write access is attempted, access terminates with an error response and no peripheral access initiates. 0 This peripheral allows write accesses. 1 This peripheral is write protected.
7 TP1	Trusted Protect Determines whether the peripheral allows accesses from an untrusted master. When this bit is set and an access is attempted by an untrusted master, the access terminates with an error response and no peripheral access initiates.

Table continues on the next page...

AIPS_PACR_n field descriptions (continued)

Field	Description
	0 Accesses from an untrusted master are allowed. 1 Accesses from an untrusted master are not allowed.
8 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
9 SP2	Supervisor Protect Determines whether the peripheral requires supervisor privilege level for accesses. When this field is set, the master privilege level must indicate the supervisor access attribute, and the MPRx[MPL _n] control field for the master must be set. If not, access terminates with an error response and no peripheral access initiates. 0 This peripheral does not require supervisor privilege level for accesses. 1 This peripheral requires supervisor privilege level for accesses.
10 WP2	Write Protect Determines whether the peripheral allows write access. When this bit is set and a write access is attempted, access terminates with an error response and no peripheral access initiates. 0 This peripheral allows write accesses. 1 This peripheral is write protected.
11 TP2	Trusted Protect Determines whether the peripheral allows accesses from an untrusted master. When this field is set and an access is attempted by an untrusted master, the access terminates with an error response and no peripheral access initiates. 0 Accesses from an untrusted master are allowed. 1 Accesses from an untrusted master are not allowed.
12 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
13 SP3	Supervisor Protect Determines whether the peripheral requires supervisor privilege level for access. When this bit is set, the master privilege level must indicate the supervisor access attribute, and the MPRx[MPL _n] control bit for the master must be set. If not, access terminates with an error response and no peripheral access initiates. 0 This peripheral does not require supervisor privilege level for accesses. 1 This peripheral requires supervisor privilege level for accesses.
14 WP3	Write Protect Determines whether the peripheral allows write accesses. When this field is set and a write access is attempted, access terminates with an error response and no peripheral access initiates. 0 This peripheral allows write accesses. 1 This peripheral is write protected.
15 TP3	Trusted Protect Determines whether the peripheral allows accesses from an untrusted master. When this bit is set and an access is attempted by an untrusted master, the access terminates with an error response and no peripheral access initiates.

Table continues on the next page...

AIPS_PACR_n field descriptions (continued)

Field	Description
	0 Accesses from an untrusted master are allowed. 1 Accesses from an untrusted master are not allowed.
16 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
17 SP4	Supervisor Protect Determines whether the peripheral requires supervisor privilege level for accesses. When this field is set, the master privilege level must indicate the supervisor access attribute, and the MPRx[MPL _n] control field for the master must be set. If not, access terminates with an error response and no peripheral access initiates. 0 This peripheral does not require supervisor privilege level for accesses. 1 This peripheral requires supervisor privilege level for accesses.
18 WP4	Write Protect Determines whether the peripheral allows write access. When this bit is set and a write access is attempted, access terminates with an error response and no peripheral access initiates. 0 This peripheral allows write accesses. 1 This peripheral is write protected.
19 TP4	Trusted Protect Determines whether the peripheral allows accesses from an untrusted master. When this field is set and an access is attempted by an untrusted master, the access terminates with an error response and no peripheral access initiates. 0 Accesses from an untrusted master are allowed. 1 Accesses from an untrusted master are not allowed.
20 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
21 SP5	Supervisor Protect Determines whether the peripheral requires supervisor privilege level for accesses. When this field is set, the master privilege level must indicate the supervisor access attribute, and the MPRx[MPL _n] control field for the master must be set. If not, access terminates with an error response and no peripheral access initiates. 0 This peripheral does not require supervisor privilege level for accesses. 1 This peripheral requires supervisor privilege level for accesses.
22 WP5	Write Protect Determines whether the peripheral allows write accesses. When this field is set and a write access is attempted, access terminates with an error response and no peripheral access initiates. 0 This peripheral allows write accesses. 1 This peripheral is write protected.
23 TP5	Trusted Protect Determines whether the peripheral allows accesses from an untrusted master. When this field is set and an access is attempted by an untrusted master, the access terminates with an error response and no peripheral access initiates.

Table continues on the next page...

AIPS_PACR_n field descriptions (continued)

Field	Description
	0 Accesses from an untrusted master are allowed. 1 Accesses from an untrusted master are not allowed.
24 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
25 SP6	Supervisor Protect Determines whether the peripheral requires supervisor privilege level for accesses. When this field is set, the master privilege level must indicate the supervisor access attribute, and the MPRx[MPL _n] control field for the master must be set. If not, access terminates with an error response and no peripheral access initiates. 0 This peripheral does not require supervisor privilege level for accesses. 1 This peripheral requires supervisor privilege level for accesses.
26 WP6	Write Protect Determines whether the peripheral allows write accesses. When this field is set and a write access is attempted, access terminates with an error response and no peripheral access initiates. 0 This peripheral allows write accesses. 1 This peripheral is write protected.
27 TP6	Trusted Protect Determines whether the peripheral allows accesses from an untrusted master. When this field is set and an access is attempted by an untrusted master, the access terminates with an error response and no peripheral access initiates. 0 Accesses from an untrusted master are allowed. 1 Accesses from an untrusted master are not allowed.
28 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
29 SP7	Supervisor Protect Determines whether the peripheral requires supervisor privilege level for accesses. When this field is set, the master privilege level must indicate the supervisor access attribute, and the MPRx[MPL _n] control field for the master must be set. If not, access terminates with an error response and no peripheral access initiates. 0 This peripheral does not require supervisor privilege level for accesses. 1 This peripheral requires supervisor privilege level for accesses.
30 WP7	Write Protect Determines whether the peripheral allows write accesses. When this field is set and a write access is attempted, access terminates with an error response and no peripheral access initiates. 0 This peripheral allows write accesses. 1 This peripheral is write protected.
31 TP7	Trusted Protect Determines whether the peripheral allows accesses from an untrusted master. When this field is set and an access is attempted by an untrusted master, the access terminates with an error response and no peripheral access initiates.

Table continues on the next page...

AIPS_PACR_n field descriptions (continued)

Field	Description
0	Accesses from an untrusted master are allowed.
1	Accesses from an untrusted master are not allowed.

11.3.4 Off-Platform Peripheral Access Control Register (AIPS_OPACR_n)

Each OPACR register consists of eight 4-bit OPACR fields. Each OPACR field defines the access levels for a particular off-platform peripheral. The peripheral assignment to each OPACR field is defined by the memory map slot of the peripheral. See the chip-specific AIPS information for the field assignment of a particular peripheral.

Every OPACR field to which no peripheral is assigned is reserved. Reads to reserved locations return zeros, and writes are ignored.

The following table shows the location of each peripheral slot's OPACR field in the OPACR registers.

Offset	Register	[0:3]	[4:7]	[8:11]	[12:15]	[16:19]	[20:23]	[24:27]	[28:31]
0x40	OPACRA	OPACR0	OPACR1	OPACR2	OPACR3	OPACR4	OPACR5	OPACR6	OPACR7
0x44	OPACRB	OPACR8	OPACR9	OPACR10	OPACR11	OPACR12	OPACR13	OPACR14	OPACR15
0x48	OPACRC	OPACR16	OPACR17	OPACR18	OPACR19	OPACR20	OPACR21	OPACR22	OPACR23
0x4C	OPACRD	OPACR24	OPACR25	OPACR26	OPACR27	OPACR28	OPACR29	OPACR30	OPACR31
0x50	OPACRE	OPACR32	OPACR33	OPACR34	OPACR35	OPACR36	OPACR37	OPACR38	OPACR39
0x54	OPACRF	OPACR40	OPACR41	OPACR42	OPACR43	OPACR44	OPACR45	OPACR46	OPACR47
0x58	OPACRG	OPACR48	OPACR49	OPACR50	OPACR51	OPACR52	OPACR53	OPACR54	OPACR55
0x5C	OPACRH	OPACR56	OPACR57	OPACR58	OPACR59	OPACR60	OPACR61	OPACR62	OPACR63

Address: 0h base + 40h offset + (4d × i), where i=0d to 7d

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	SP0	WP0	TP0	0	SP1	WP1	TP1	0	SP2	WP2	TP2	0	SP3	WP3	TP3
W																
Reset	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	SP4	WP4	TP4	0	SP5	WP5	TP5	0	SP6	WP6	TP6	0	SP7	WP7	TP7
W																
Reset	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*

* Notes:

- The reset value is chip-dependent and can be found in the AIPS chip-specific information.

AIPS_OPACR_n field descriptions

Field	Description
0 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
1 SP0	Supervisor Protect Determines whether the peripheral requires supervisor privilege level for accesses. When this field is set, the master privilege level must indicate the supervisor access attribute, and the MPRx[MPL _n] control field for the master must be set. If not, access terminates with an error response and no peripheral access initiates. 0 This peripheral does not require supervisor privilege level for accesses. 1 This peripheral requires supervisor privilege level for accesses.
2 WP0	Write Protect Determines whether the peripheral allows write accesses. When this field is set and a write access is attempted, access terminates with an error response and no peripheral access initiates. 0 This peripheral allows write accesses. 1 This peripheral is write protected.
3 TP0	Trusted Protect Determines whether the peripheral allows accesses from an untrusted master. When this bit is set and an access is attempted by an untrusted master, the access terminates with an error response and no peripheral access initiates. 0 Accesses from an untrusted master are allowed. 1 Accesses from an untrusted master are not allowed.
4 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
5 SP1	Supervisor Protect Determines whether the peripheral requires supervisor privilege level for access. When this field is set, the master privilege level must indicate the supervisor access attribute, and the MPRx[MPL _n] control field for the master must be set. If not, access terminates with an error response and no peripheral access initiates. 0 This peripheral does not require supervisor privilege level for accesses. 1 This peripheral requires supervisor privilege level for accesses.
6 WP1	Write Protect Determines whether the peripheral allows write accesses. When this field is set and a write access is attempted, access terminates with an error response and no peripheral access initiates. 0 This peripheral allows write accesses. 1 This peripheral is write protected.
7 TP1	Trusted Protect Determines whether the peripheral allows accesses from an untrusted master. When this field is set and an access is attempted by an untrusted master, the access terminates with an error response and no peripheral access initiates. 0 Accesses from an untrusted master are allowed. 1 Accesses from an untrusted master are not allowed.

Table continues on the next page...

AIPS_OPACR_n field descriptions (continued)

Field	Description
8 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
9 SP2	Supervisor Protect Determines whether the peripheral requires supervisor privilege level for access. When this bit is set, the master privilege level must indicate the supervisor access attribute, and the MPRx[MPL _n] control bit for the master must be set. If not, access terminates with an error response and no peripheral access initiates. 0 This peripheral does not require supervisor privilege level for accesses. 1 This peripheral requires supervisor privilege level for accesses.
10 WP2	Write Protect Determines whether the peripheral allows write accesses. When this field is set and a write access is attempted, access terminates with an error response and no peripheral access initiates. 0 This peripheral allows write accesses. 1 This peripheral is write protected.
11 TP2	Trusted Protect Determines whether the peripheral allows accesses from an untrusted master. When this bit is set and an access is attempted by an untrusted master, the access terminates with an error response and no peripheral access initiates. 0 Accesses from an untrusted master are allowed. 1 Accesses from an untrusted master are not allowed.
12 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
13 SP3	Supervisor Protect Determines whether the peripheral requires supervisor privilege level for accesses. When this field is set, the master privilege level must indicate the supervisor access attribute, and the MPRx[MPL _n] control field for the master must be set. If not, access terminates with an error response and no peripheral access initiates. 0 This peripheral does not require supervisor privilege level for accesses. 1 This peripheral requires supervisor privilege level for accesses.
14 WP3	Write Protect Determines whether the peripheral allows write access. When this bit is set and a write access is attempted, access terminates with an error response and no peripheral access initiates. 0 This peripheral allows write accesses. 1 This peripheral is write protected.
15 TP3	Trusted Protect Determines whether the peripheral allows accesses from an untrusted master. When this field is set and an access is attempted by an untrusted master, the access terminates with an error response and no peripheral access initiates. 0 Accesses from an untrusted master are allowed. 1 Accesses from an untrusted master are not allowed.

Table continues on the next page...

AIPS_OPACR_n field descriptions (continued)

Field	Description
16 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
17 SP4	Supervisor Protect Determines whether the peripheral requires supervisor privilege level for access. When this bit is set, the master privilege level must indicate the supervisor access attribute, and the MPRx[MPL _n] control bit for the master must be set. If not, access terminates with an error response and no peripheral access initiates. 0 This peripheral does not require supervisor privilege level for accesses. 1 This peripheral requires supervisor privilege level for accesses.
18 WP4	Write Protect Determines whether the peripheral allows write accesses. When this field is set and a write access is attempted, access terminates with an error response and no peripheral access initiates. 0 This peripheral allows write accesses. 1 This peripheral is write protected.
19 TP4	Trusted Protect Determines whether the peripheral allows accesses from an untrusted master. When this bit is set and an access is attempted by an untrusted master, the access terminates with an error response and no peripheral access initiates. 0 Accesses from an untrusted master are allowed. 1 Accesses from an untrusted master are not allowed.
20 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
21 SP5	Supervisor Protect Determines whether the peripheral requires supervisor privilege level for accesses. When this field is set, the master privilege level must indicate the supervisor access attribute, and the MPRx[MPL _n] control field for the master must be set. If not, access terminates with an error response and no peripheral access initiates. 0 This peripheral does not require supervisor privilege level for accesses. 1 This peripheral requires supervisor privilege level for accesses.
22 WP5	Write Protect Determines whether the peripheral allows write accesses. When this field is set and a write access is attempted, access terminates with an error response and no peripheral access initiates. 0 This peripheral allows write accesses. 1 This peripheral is write protected.
23 TP5	Trusted Protect Determines whether the peripheral allows accesses from an untrusted master. When this field is set and an access is attempted by an untrusted master, the access terminates with an error response and no peripheral access initiates. 0 Accesses from an untrusted master are allowed. 1 Accesses from an untrusted master are not allowed.

Table continues on the next page...

AIPS_OPACR_n field descriptions (continued)

Field	Description
24 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
25 SP6	Supervisor Protect Determines whether the peripheral requires supervisor privilege level for accesses. When this field is set, the master privilege level must indicate the supervisor access attribute, and the MPRx[MPL _n] control field for the master must be set. If not, access terminates with an error response and no peripheral access initiates. 0 This peripheral does not require supervisor privilege level for accesses. 1 This peripheral requires supervisor privilege level for accesses.
26 WP6	Write Protect Determines whether the peripheral allows write accesses. When this field is set and a write access is attempted, access terminates with an error response and no peripheral access initiates. 0 This peripheral allows write accesses. 1 This peripheral is write protected.
27 TP6	Trusted Protect Determines whether the peripheral allows accesses from an untrusted master. When this field is set and an access is attempted by an untrusted master, the access terminates with an error response and no peripheral access initiates. 0 Accesses from an untrusted master are allowed. 1 Accesses from an untrusted master are not allowed.
28 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
29 SP7	Supervisor Protect Determines whether the peripheral requires supervisor privilege level for accesses. When this field is set, the master privilege level must indicate the supervisor access attribute, and the MPRx[MPL _n] control field for the master must be set. If not, access terminates with an error response and no peripheral access initiates. 0 This peripheral does not require supervisor privilege level for accesses. 1 This peripheral requires supervisor privilege level for accesses.
30 WP7	Write Protect Determines whether the peripheral allows write accesses. When this field is set and a write access is attempted, access terminates with an error response and no peripheral access initiates. 0 This peripheral allows write accesses. 1 This peripheral is write protected.
31 TP7	Trusted Protect Determines whether the peripheral allows accesses from an untrusted master. When this field is set and an access is attempted by an untrusted master, the access terminates with an error response and no peripheral access initiates. 0 Accesses from an untrusted master are allowed. 1 Accesses from an untrusted master are not allowed.

11.4 Functional description

The peripheral bridge functions as a bus protocol translator between the crossbar switch and the slave peripheral bus.

Support is provided for generating a pair of 32-bit slave accesses when performing certain 64-bit peripheral accesses.

The peripheral bridge manages all transactions destined for the attached slave devices and generates select signals for modules on the peripheral bus by decoding accesses within the attached address space.

11.4.1 Access support

All combinations of access size and peripheral data port width are supported. An access that is larger than the target peripheral's data width will be decomposed to multiple, smaller accesses. Bus decomposition is terminated by a transfer error caused by an access to an empty register area.

Chapter 12

Platform Configuration Module (PCM)

12.1 PCM memory map and register descriptions

The Platform Configuration Module contains three miscellaneous configuration registers for the chip. Currently, the configuration registers are related to the operation of the FEC and intelligent bus bridging gasket. The module is mapped to AIPS_0 (PBRIDGE_B) on-platform slot 27 with a base address of FFF6_C000h.

NOTE

These registers can be accessed only in supervisor mode.

PCM memory map

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/page
0	FEC Burst Optimization Master Control Register (PCM_FBOMCR)	32	R/W	0000_0000h	12.1.1/490
4	Bus Bridge Configuration Register 1 (PCM_IAHB_BE1)	32	R/W	0303_0303h	12.1.2/491
8	Bus Bridge Configuration Register 2 (PCM_IAHB_BE2)	32	R/W	0303_0303h	12.1.3/494

12.1.1 FEC Burst Optimization Master Control Register (PCM_FBOMCR)

This register controls FEC burst optimization behavior on the system bus.

Address: 0h base + 0h offset = 0h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0					ACCERR	WBEN	RBEN	FXSBE[7:0]							
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

PCM_FBOMCR field descriptions

Field	Description
0–20 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
21 ACCERR	Accumulate Error This field determines whether an error response for the first half of the write burst is accumulated to the second half of the write burst or discarded. To complete the burst, the FEC interface to the system bus responds by indicating that the first half of the burst completed without error before it actually writes the data so that it can fetch the second half of the write data from the FIFO. When actually written onto the system bus, the first half of the write burst can have an error. Because this half initially responded without an error to the FIFO, the error is discarded or accumulated with the error response for the second half of the burst. 0 Any error to the first half of the write burst is discarded. 1 Any actual error response to the first half of the write burst is accumulated in the second half's response. In other words, an error response to the first half is seen in the response to the second half, even if the second half does not contain an error.
22 WBEN	Global write burst enable to XBAR slave port designated by FXSBE _n 0 Write bursting to all XBAR slave ports is disabled. 1 Write bursting is enabled to any XBAR slave port whose FXSBE _n bit is 1.
23 RBEN	Global read burst enable from XBAR slave port designated by FXSBE _n 0 Read bursting from all XBAR slave ports is disabled. 1 Read bursting is enabled from any XBAR slave port whose FXSBE _n bit is 1.

Table continues on the next page...

PCM_FBOMCR field descriptions (continued)

Field	Description
24–31 FXSBE[7:0]	FEC XBAR slave burst enable This field enables bursting by the FEC interface to the XBAR slave port controlled by each FXSBE_n bit. - When a particular FXSBE_n bit is 1, the XBAR slave port enabled by that bit can support the bursts allowed by RBEN and WBEN. RBEN enables read bursts from the XBAR slave port, and WBEN enables write bursts to the XBAR slave port. - When a particular FXSBE_n bit is 0, the FEC interface does not burst to the XBAR slave port controlled by that FXSBE_n bit. FXSBE_n assignments to XBAR slave ports are as follows: FXSBE0 = Flash memory FXSBE1 = EBI FXSBE2 = RAM FXSBE3 = reserved FXSBE4 = reserved FXSBE5 = reserved FXSBE6 = AIPS_1 (PBRIDGE_A) FXSBE7 = AIPS_0 (PBRIDGE_B)

12.1.2 Bus Bridge Configuration Register 1 (PCM_IAHB_BE1)

Address: 0h base + 4h offset = 4h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0					PRE_CORE1_	BRE_CORE1_	BWE_CORE1_	0					PRE_CORE1_I	BRE_CORE1_I	BWE_CORE1_I
W																
Reset	0	0	0	0	0	0	1	1	0	0	0	0	0	0	1	1

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0					PRE_CORE0_	BRE_CORE0_	BWE_CORE0_	0					PRE_CORE0_I	BRE_CORE0_I	BWE_CORE0_I
W																
Reset	0	0	0	0	0	0	1	1	0	0	0	0	0	0	1	1

PCM_IAHB_BE1 field descriptions

Field	Description
0–4 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

Table continues on the next page...

PCM_IAHB_BE1 field descriptions (continued)

Field	Description
5 PRE_CORE1_D	Pending Read Enable Core1 Data This bit controls the bus gasket's handling of pending read transactions. 0 Pending reads are disabled. 1 Pending reads are enabled.
6 BRE_CORE1_D	Burst Read Enable Core1 Data This bit controls the bus gasket's handling of burst read transactions. 0 Burst reads are converted into a series of single transactions on the slave side of the gasket. 1 Burst reads are optimized for best system performance.
7 BWE_CORE1_D	Burst Write Enable Core1 Data This bit controls the bus gasket's handling of burst write transactions. 0 Burst writes are converted into a series of single transactions on the slave side of the gasket. 1 Burst writes are optimized for best system performance. Note this setting treats writes as "imprecise" such that an error response on any beat of the burst is reported on the last beat.
8–12 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
13 PRE_CORE1_I	Pending Read Enable Core1 Instruction This bit controls the bus gasket's handling of pending read transactions. 0 Pending reads are disabled. 1 Pending reads are enabled.
14 BRE_CORE1_I	Burst Read Enable Core1 Instruction This bit controls the bus gasket's handling of burst read transactions. 0 Burst reads are converted into a series of single transactions on the slave side of the gasket. 1 Burst reads are optimized for best system performance.
15 BWE_CORE1_I	Burst Write Enable Core1 Instruction This bit controls the bus gasket's handling of burst write transactions. 0 Burst writes are converted into a series of single transactions on the slave side of the gasket. 1 Burst writes are optimized for best system performance. Note this setting treats writes as "imprecise" such that an error response on any beat of the burst is reported on the last beat.
16–20 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
21 PRE_CORE0_D	Pending Read Enable Core0 Data This bit controls the bus gasket's handling of pending read transactions. 0 Pending reads are disabled 1 Pending reads are enabled.
22 BRE_CORE0_D	Burst Read Enable Core0 Data This bit controls the bus gasket's handling of burst read transactions.

Table continues on the next page...

PCM_IAHB_BE1 field descriptions (continued)

Field	Description
	0 Burst reads are converted into a series of single transactions on the slave side of the gasket. 1 Burst reads are optimized for best system performance.
23 BWE_CORE0_D	Burst Write Enable Core0 Data This bit controls the bus gasket's handling of burst write transactions. 0 Burst writes are converted into a series of single transactions on the slave side of the gasket. 1 Burst writes are optimized for best system performance. Note this setting treats writes as "imprecise" such that an error response on any beat of the burst is reported on the last beat.
24–28 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
29 PRE_CORE0_I	Pending Read Enable Core0 Instruction This bit controls the bus gasket's handling of pending read transactions. 0 Pending reads are disabled. 1 Pending reads are enabled.
30 BRE_CORE0_I	Burst Read Enable Core0 Instruction This bit controls the bus gasket's handling of burst read transactions. 0 Burst reads are converted into a series of single transactions on the slave side of the gasket. 1 Burst reads are optimized for best system performance.
31 BWE_CORE0_I	Burst Write Enable Core0 Instruction This bit controls the bus gasket's handling of burst write transactions. 0 Burst writes are converted into a series of single transactions on the slave side of the gasket. 1 Burst writes are optimized for best system performance. Note this setting treats writes as "imprecise" such that an error response on any beat of the burst is reported on the last beat.

12.1.3 Bus Bridge Configuration Register 2 (PCM_IAHB_BE2)

Address: 0h base + 8h offset = 8h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0					PRE_FEC	BRE_FEC	BWE_FEC	0					PRE_M6	BRE_M6	BWE_M6
W																
Reset	0	0	0	0	0	0	1	1	0	0	0	0	0	0	1	1
Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0					PRE_DMA_B	BRE_DMA_B	BWE_DMA_B	0					PRE_DMA_A	BRE_DMA_A	BWE_DMA_A
W																
Reset	0	0	0	0	0	0	1	1	0	0	0	0	0	0	1	1

PCM_IAHB_BE2 field descriptions

Field	Description
0–4 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
5 PRE_FEC	Pending Read Enable FEC This bit controls the bus gasket's handling of pending read transactions. 0 Pending reads are disabled. 1 Pending reads are enabled.
6 BRE_FEC	Burst Read Enable FEC This bit controls the bus gasket's handling of burst read transactions. 0 Burst reads are converted into a series of single transactions on the slave side of the gasket. 1 Burst reads are optimized for best system performance.
7 BWE_FEC	Burst Write Enable FEC This bit controls the bus gasket's handling of burst write transactions. 0 Burst writes are converted into a series of single transactions on the slave side of the gasket. 1 Burst writes are optimized for best system performance. Note this setting treats writes as "imprecise" such that an error response on any beat of the burst is reported on the last beat.
8–12 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
13 PRE_M6	Pending Read Enable Master Port 6 Concentrator This bit controls the bus gasket's handling of pending read transactions.

Table continues on the next page...

PCM_IAHB_BE2 field descriptions (continued)

Field	Description
	0 Pending reads are disabled. 1 Pending reads are enabled.
14 BRE_M6	Burst Read Enable Master Port 6 Concentrator This bit controls the bus gasket's handling of burst read transactions. 0 Burst reads are converted into a series of single transactions on the slave side of the gasket. 1 Burst reads are optimized for best system performance.
15 BWE_M6	Burst Write Enable Master Port 6 Concentrator This bit controls the bus gasket's handling of burst write transactions. 0 Burst writes are converted into a series of single transactions on the slave side of the gasket. 1 Burst writes are optimized for best system performance. Note this setting treats writes as "imprecise" such that an error response on any beat of the burst is reported on the last beat.
16–20 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
21 PRE_DMA_B	Pending Read Enable eDMA_B This bit controls the bus gasket's handling of pending read transactions. 0 Pending reads are disabled 1 Pending reads are enabled.
22 BRE_DMA_B	Burst Read Enable eDMA_B This bit controls the bus gasket's handling of burst read transactions. 0 Burst reads are converted into a series of single transactions on the slave side of the gasket. 1 Burst reads are optimized for best system performance.
23 BWE_DMA_B	Burst Write Enable eDMA_B This bit controls the bus gasket's handling of burst write transactions. 0 Burst writes are converted into a series of single transactions on the slave side of the gasket. 1 Burst writes are optimized for best system performance. Note this setting treats writes as "imprecise" such that an error response on any beat of the burst is reported on the last beat.
24–28 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
29 PRE_DMA_A	Pending Read Enable eDMA_A This bit controls the bus gasket's handling of pending read transactions. 0 Pending reads are disabled. 1 Pending reads are enabled.
30 BRE_DMA_A	Burst Read Enable eDMA_A This bit controls the bus gasket's handling of burst read transactions. 0 Burst reads are converted into a series of single transactions on the slave side of the gasket. 1 Burst reads are optimized for best system performance.

Table continues on the next page...



PCM_IAHB_BE2 field descriptions (continued)

Field	Description
31 BWE_DMA_A	Burst Write Enable eDMA_A This bit controls the bus gasket’s handling of burst write transactions. 0 Burst writes are converted into a series of single transactions on the slave side of the gasket. 1 Burst writes are optimized for best system performance. Note this setting treats writes as “imprecise” such that an error response on any beat of the burst is reported on the last beat.

Chapter 13

Semaphores (SEMA4)

13.1 Introduction

The module provides the hardware support needed in multi-core systems for implementing semaphores and provide a simple mechanism to achieve "lock/unlock" operations via a single write access. This approach eliminates architecture-specific implementations like atomic (indivisible) read-modify-write instructions or reservation mechanisms. The result is an architecture-neutral solution that provides hardware-enforced gates as well as other useful system functions related to the gating mechanisms.

13.1.1 Multi-Core Programming 101: Software Gates

Multi-processor systems require a function that can be used to safely and easily provide a locking mechanism which is then used by system software to control access to shared data structures, shared hardware resources, etc. These "gating" mechanisms are used by the software to serialize (and synchronize) writes to shared data and/or resources to prevent race conditions and preserve memory coherency between different processes and processors.

Consider the following description of a typical use-case. Processor X enters a section of code where shared data values are to be updated. It must first acquire a semaphore; this can be considered "locking (or closing) a software gate." Once the gate has been locked, a properly-architected software system does not allow other processes (or processors) to execute the same code segment or modify the shared data structure protected by the gate, that is, other processes/processors are locked out. Many software implementations include a *spin-wait loop* within the lock function until the locking of the gate is accomplished. Once the lock has been obtained, Processor X continues execution and updates the data values protected by the particular lock. Once the updates are complete, Processor X unlocks (or opens) the software gate, allowing other processes/processors access to the updated data values.

There are three important rules that must be followed for a correctly-implemented system solution:

1. All writes to shared data values or shared hardware resources must be protected by a "gate" variable.
2. Once a processor locks a gate, accesses to the shared data or resources by other processes/processors must be blocked. This is enforced by software conventions.
3. The processor that locks a particular gate is the only processor that can open (unlock) that gate.

Information in the hardware gate identifying the locking processor can be extremely useful for system-level debugging.

The Hennessy/Patterson text on computer architecture offers the following description of software gating.

"One of the major requirements of a shared-memory architecture multiprocessor is being able to coordinate processes that are working on a common task. Typically, a programmer will use *lock variables* to synchronize the processes.

The difficulty for the architect of a multiprocessor is to provide a mechanism to decide which processor gets the lock and to provide the operation that locks a variable. Arbitration is easy for shared-bus multiprocessors, since the bus is the only path to memory. The processor that gets the bus locks out all the other processors from memory. If the CPU and bus provide an atomic swap operation, programmers can create locks with the proper semantics. The adjective *atomic* is key, for it means that a processor can both read a location **and** set it to the locked value in the same bus operation, preventing any other processor from reading or writing memory." [Hennessy/Patterson, *Computer Architecture: A Quantitative Approach*, ppg. 471-472]

The classic text continues with a description of the steps required to lock/unlock a variable using an *atomic swap instruction*.

"Assume that 0 means unlocked and 1 means locked. A processor first reads the lock variable to test its state. A processor keeps reading and testing until the value indicates that the lock is unlocked. The processor then races against all other processes that were similarly "spin waiting" to see who can lock the variable first. All processes use a swap instruction that reads the old value and stores a 1 into the lock variable. The single winner will see the 0, and the losers will see a 1 that was placed there by the winner. (The losers will continue to set the variable to the locked value, but that doesn't matter.) The winning processor executes the code after the lock and then stores a 0 into the lock when it exits,

starting the race all over again. Testing the old value and then setting to a new value is why the atomic swap instruction is called *test and set* in some instruction sets."

[Hennessy/Patterson, *Computer Architecture: A Quantitative Approach*, ppg. 472-473]

For multi-core SoCs developed in the Microcontroller Division, a hardware-enforced semaphore implementation as a platform IPS module is used for the following reasons:

- A smaller module size versus a platform reservation monitor for PowerPC-based designs and it does not introduce any timing-critical paths, for example, the AHB system bus address phase cycle like a reservation monitor
- Provides a more robust implementation with features not possible in a reservation-only approach
- Provides a more generic, architecture-neutral solution, applicable across *all* the processor families supported by the Microcontroller Division

The sole drawback to a hardware-based semaphore module is the limited number of semaphores versus the "infinite" number that can be supported with PowerPC reservation instructions.

13.1.2 Overview

The IPS_Semaphores module provides a platform IPS slave device which implements 16 hardware-enforced gates with the following features:

- Module definition supports 16 hardware-enforced gates in a dual-processor configuration, where cp0 is core processor 0 and cp1 is core processor 1
 - Hardware gates appear as a 16-entry byte-size array with read and write accesses
 - Processors lock gates by writing "processor_number+1" to the appropriate gate and must read back the gate value to verify the lock operation was successful
 - Once locked, the gate is unlocked by a write of zeroes from the locking processor
- Optional interrupt notification after a failed lock write provides a mechanism to indicate when the gate is unlocked

- Secure reset mechanisms are supported to clear the contents of individual semaphore gates or notification logic, as well as a clear_all capability
- Programming model allocates memory space to support up to 8 processors and up to 64 gates

A simplified block diagram of the Semaphores module is shown in [Figure 13-1](#). In the diagram, the register blocks named gate0, gate1,..., gate 15 include the finite state machines implementing the semaphore gates plus the interrupt notification logic.

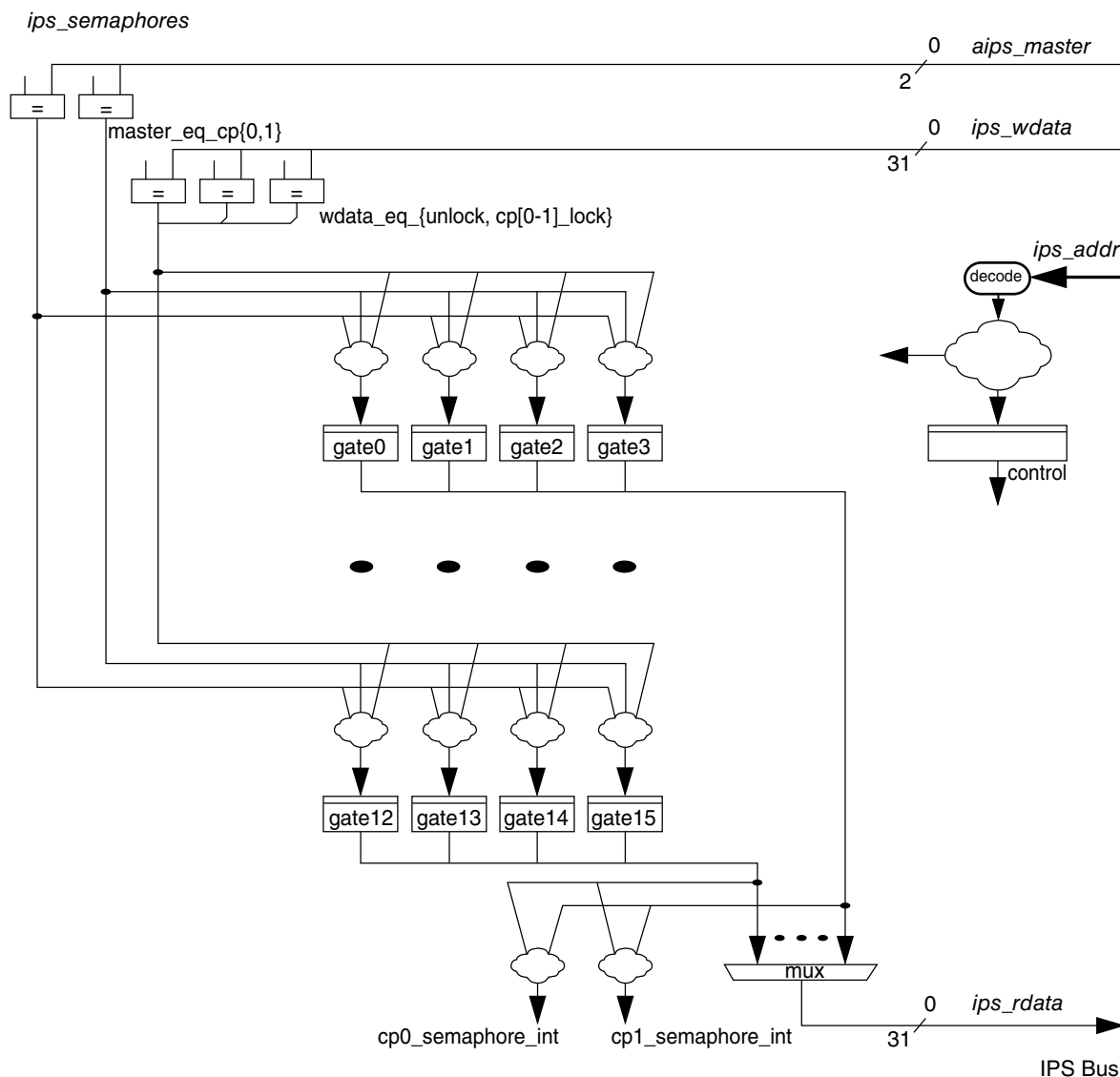


Figure 13-1. IPS_Semaphores Block Diagram

13.1.3 Features

The Semaphores module implements hardware-enforced semaphores as an IPS-mapped slave peripheral device. The feature set includes:

- Support for 16 hardware-enforced gates in a dual-processor configuration
 - Each hardware gate appears as a 3-state, 2-bit state machine, with all 16 gates mapped as a byte-size array
 - 3-state implementation
 - if gate = 0b00, then state = unlocked
 - if gate = 0b01, then state = locked by processor 0
 - if gate = 0b10, then state = locked by processor 1
 - Uses the bus master number as a reference attribute plus the specified data patterns to validate all write operations
 - Once locked, the gate can (and must) be unlocked by a write of zeroes from the locking processor
- Optional interrupt notification after a failed lock write provides a mechanism to indicate when the gate is unlocked
- Secure reset mechanisms are supported to clear the contents of individual gates or notification logic, as well as a clear_all capability
- Memory-mapped IPS slave peripheral platform module
 - Interface to the IPS bus for programming-model accesses
 - Two outputs (one per processor) for interrupt notification of failed lock writes

13.1.4 Modes of Operation

The Semaphores module does not support any special modes of operation. As a slave peripheral memory-mapped device located on the platform's IPS slave bus, it responds based strictly on the memory addresses of the connected bus. The IPS bus is used to access the Semaphores ' programming model.

13.2 External Signal Description

The Semaphores module does not include any external interfaces.

13.3 Memory map and register definition

The Semaphores module provides an IPS programming model mapped to an SPP-standard on-platform 16 KB space. The description here specifies a dual-core configuration with 16 semaphore gates. All the register names are prefixed with "Sema4" as an abbreviation for the full module name.

The programming model is referenced using 8-, 16- and 32-bit accesses. Reads can use any reference size, while writes are generally restricted to the size of the register. Exceptions to the write size restrictions are detailed in the individual register descriptions. Attempted references using inappropriate access sizes, to undefined (reserved) addresses, or with a non-supported access type (for example, a write to a read-only register) generate an IPS error termination.

Finally, the programming model allocates space for a definition with up to 64 gates and up to 8 processor cores, even though this definition is considerably larger than any currently-planned module implementations. The number of gates and supported processor cores are independent; there is no relationship between these two system variables.

The 16 KB Semaphores programming model map is shown in the following table.

SEMA4 memory map

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
0	Semaphores Gate 0 Register (SEMA4_Gate00)	8	R/W	00h	13.3.1/503
1	Semaphores Gate 1 Register (SEMA4_Gate01)	8	R/W	00h	13.3.2/504
2	Semaphores Gate 2 Register (SEMA4_Gate02)	8	R/W	00h	13.3.3/505
3	Semaphores Gate 3 Register (SEMA4_Gate03)	8	R/W	00h	13.3.4/506
4	Semaphores Gate 4 Register (SEMA4_Gate04)	8	R/W	00h	13.3.5/507
5	Semaphores Gate 5 Register (SEMA4_Gate05)	8	R/W	00h	13.3.6/508
6	Semaphores Gate 6 Register (SEMA4_Gate06)	8	R/W	00h	13.3.7/509
7	Semaphores Gate 7 Register (SEMA4_Gate07)	8	R/W	00h	13.3.8/510
8	Semaphores Gate 8 Register (SEMA4_Gate08)	8	R/W	00h	13.3.9/511
9	Semaphores Gate 9 Register (SEMA4_Gate09)	8	R/W	00h	13.3.10/512
A	Semaphores Gate 10 Register (SEMA4_Gate10)	8	R/W	00h	13.3.11/513
B	Semaphores Gate 11 Register (SEMA4_Gate11)	8	R/W	00h	13.3.12/514
C	Semaphores Gate 12 Register (SEMA4_Gate12)	8	R/W	00h	13.3.13/515

Table continues on the next page...

SEMA4 memory map (continued)

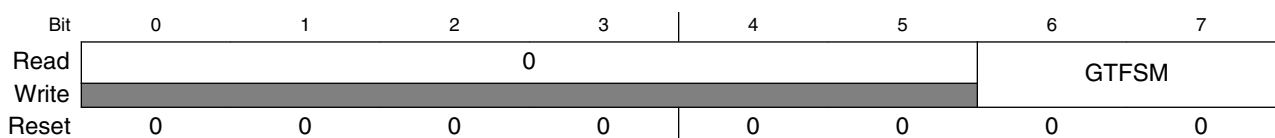
Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
D	Semaphores Gate 13 Register (SEMA4_Gate13)	8	R/W	00h	13.3.14/ 516
E	Semaphores Gate 14 Register (SEMA4_Gate14)	8	R/W	00h	13.3.15/ 517
F	Semaphores Gate 15 Register (SEMA4_Gate15)	8	R/W	00h	13.3.16/ 518
40	Semaphores Processor n IRQ Notification Enable (SEMA4_CP0INE)	16	R/W	0000h	13.3.17/ 519
48	Semaphores Processor n IRQ Notification Enable (SEMA4_CP1INE)	16	R/W	0000h	13.3.17/ 519
80	Semaphores Processor n IRQ Notification (SEMA4_CP0NTF)	16	R	0000h	13.3.18/ 521
88	Semaphores Processor n IRQ Notification (SEMA4_CP1NTF)	16	R	0000h	13.3.18/ 521
100	Semaphores (Secure) Reset Gate n (SEMA4_RSTGT)	16	R/W	0000h	13.3.19/ 522
104	Semaphores (Secure) Reset IRQ Notification (SEMA4_RSTNTF)	16	R/W	0000h	13.3.20/ 524

13.3.1 Semaphores Gate 0 Register (SEMA4_Gate00)

Each semaphore gate is implemented in a 2-bit finite state machine, right-justified in a byte data structure. The hardware uses the bus master number in conjunction with the data patterns to validate all attempted write operations. Only processor bus masters can modify the gate registers. Once locked, a gate can (and must) be opened (unlocked) by the locking processor core.

Multiple gate values can be read in a single access, but only a single gate can be updated via a write operation at a time. 16- and 32-bit writes to multiple gates are allowed, but the write data operand must only update the state of a single gate. A byte write data value of 0x03 is defined as "no operation" and does not affect the state of the corresponding gate register. Attempts to write multiple gates in a single aligned access with a size larger than an 8-bit (byte) reference generate an error termination and do not allow any gate state changes

Address: 0h base + 0h offset = 0h



SEMA4_Gate00 field descriptions

Field	Description
0–5 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
6–7 GTFSM	Gate Finite State Machine. Gate Finite State Machine. The hardware gate is maintained in a 3-state implementation-unlocked, locked by processor 0 or locked by processor 1. For more details, see SEMA4_GATEn Operation . NOTE: The state of the gate reflects the last processor that locked it, which can be useful during system debug. 00 The gate is unlocked (free). 01 The gate has been locked by processor 0. 10 The gate has been locked by processor 1. 11 This state encoding is never used and therefore reserved. Attempted writes of 0x03 are treated as "no operation" and do not affect the gate state machine.

13.3.2 Semaphores Gate 1 Register (SEMA4_Gate01)

Each semaphore gate is implemented in a 2-bit finite state machine, right-justified in a byte data structure. The hardware uses the bus master number in conjunction with the data patterns to validate all attempted write operations. Only processor bus masters can modify the gate registers. Once locked, a gate can (and must) be opened (unlocked) by the locking processor core.

Multiple gate values can be read in a single access, but only a single gate can be updated via a write operation at a time. 16- and 32-bit writes to multiple gates are allowed, but the write data operand must only update the state of a single gate. A byte write data value of 0x03 is defined as "no operation" and does not affect the state of the corresponding gate register. Attempts to write multiple gates in a single aligned access with a size larger than an 8-bit (byte) reference generate an error termination and do not allow any gate state changes

Address: 0h base + 1h offset = 1h

Bit	0	1	2	3	4	5	6	7
Read	0					GTFSM		
Write								
Reset	0	0	0	0	0	0	0	0

SEMA4_Gate01 field descriptions

Field	Description
0–5 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

Table continues on the next page...

SEMA4_Gate01 field descriptions (continued)

Field	Description
6–7 GTFSM	Gate Finite State Machine. Gate Finite State Machine. The hardware gate is maintained in a 3-state implementation-unlocked, locked by processor 0 or locked by processor 1. For more details, see SEMA4_GATEn Operation . NOTE: The state of the gate reflects the last processor that locked it, which can be useful during system debug. 00 The gate is unlocked (free). 01 The gate has been locked by processor 0. 10 The gate has been locked by processor 1. 11 This state encoding is never used and therefore reserved. Attempted writes of 0x03 are treated as "no operation" and do not affect the gate state machine.

13.3.3 Semaphores Gate 2 Register (SEMA4_Gate02)

Each semaphore gate is implemented in a 2-bit finite state machine, right-justified in a byte data structure. The hardware uses the bus master number in conjunction with the data patterns to validate all attempted write operations. Only processor bus masters can modify the gate registers. Once locked, a gate can (and must) be opened (unlocked) by the locking processor core.

Multiple gate values can be read in a single access, but only a single gate can be updated via a write operation at a time. 16- and 32-bit writes to multiple gates are allowed, but the write data operand must only update the state of a single gate. A byte write data value of 0x03 is defined as "no operation" and does not affect the state of the corresponding gate register. Attempts to write multiple gates in a single aligned access with a size larger than an 8-bit (byte) reference generate an error termination and do not allow any gate state changes

Address: 0h base + 2h offset = 2h

Bit	0	1	2	3	4	5	6	7
Read	0					GTFSM		
Write								
Reset	0	0	0	0	0	0	0	0

SEMA4_Gate02 field descriptions

Field	Description
0–5 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
6–7 GTFSM	Gate Finite State Machine. Gate Finite State Machine. The hardware gate is maintained in a 3-state implementation-unlocked, locked by processor 0 or locked by processor 1. For more details, see SEMA4_GATEn Operation .

Table continues on the next page...

SEMA4_Gate02 field descriptions (continued)

Field	Description
	NOTE: The state of the gate reflects the last processor that locked it, which can be useful during system debug.
00	The gate is unlocked (free).
01	The gate has been locked by processor 0.
10	The gate has been locked by processor 1.
11	This state encoding is never used and therefore reserved. Attempted writes of 0x03 are treated as "no operation" and do not affect the gate state machine.

13.3.4 Semaphores Gate 3 Register (SEMA4_Gate03)

Each semaphore gate is implemented in a 2-bit finite state machine, right-justified in a byte data structure. The hardware uses the bus master number in conjunction with the data patterns to validate all attempted write operations. Only processor bus masters can modify the gate registers. Once locked, a gate can (and must) be opened (unlocked) by the locking processor core.

Multiple gate values can be read in a single access, but only a single gate can be updated via a write operation at a time. 16- and 32-bit writes to multiple gates are allowed, but the write data operand must only update the state of a single gate. A byte write data value of 0x03 is defined as "no operation" and does not affect the state of the corresponding gate register. Attempts to write multiple gates in a single aligned access with a size larger than an 8-bit (byte) reference generate an error termination and do not allow any gate state changes

Address: 0h base + 3h offset = 3h



SEMA4_Gate03 field descriptions

Field	Description
0–5 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
6–7 GTFSM	Gate Finite State Machine. Gate Finite State Machine. The hardware gate is maintained in a 3-state implementation-unlocked, locked by processor 0 or locked by processor 1. For more details, see SEMA4_GATEn Operation . NOTE: The state of the gate reflects the last processor that locked it, which can be useful during system debug. 00 The gate is unlocked (free).

Table continues on the next page...

SEMA4_Gate03 field descriptions (continued)

Field	Description
01	The gate has been locked by processor 0.
10	The gate has been locked by processor 1.
11	This state encoding is never used and therefore reserved. Attempted writes of 0x03 are treated as "no operation" and do not affect the gate state machine.

13.3.5 Semaphores Gate 4 Register (SEMA4_Gate04)

Each semaphore gate is implemented in a 2-bit finite state machine, right-justified in a byte data structure. The hardware uses the bus master number in conjunction with the data patterns to validate all attempted write operations. Only processor bus masters can modify the gate registers. Once locked, a gate can (and must) be opened (unlocked) by the locking processor core.

Multiple gate values can be read in a single access, but only a single gate can be updated via a write operation at a time. 16- and 32-bit writes to multiple gates are allowed, but the write data operand must only update the state of a single gate. A byte write data value of 0x03 is defined as "no operation" and does not affect the state of the corresponding gate register. Attempts to write multiple gates in a single aligned access with a size larger than an 8-bit (byte) reference generate an error termination and do not allow any gate state changes

Address: 0h base + 4h offset = 4h

Bit	0	1	2	3	4	5	6	7
Read	0					GTFSM		
Write								
Reset	0	0	0	0	0	0	0	0

SEMA4_Gate04 field descriptions

Field	Description
0–5 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
6–7 GTFSM	Gate Finite State Machine. Gate Finite State Machine. The hardware gate is maintained in a 3-state implementation-unlocked, locked by processor 0 or locked by processor 1. For more details, see SEMA4_GATEn Operation . NOTE: The state of the gate reflects the last processor that locked it, which can be useful during system debug. 00 The gate is unlocked (free). 01 The gate has been locked by processor 0. 10 The gate has been locked by processor 1. 11 This state encoding is never used and therefore reserved. Attempted writes of 0x03 are treated as "no operation" and do not affect the gate state machine.

13.3.6 Semaphores Gate 5 Register (SEMA4_Gate05)

Each semaphore gate is implemented in a 2-bit finite state machine, right-justified in a byte data structure. The hardware uses the bus master number in conjunction with the data patterns to validate all attempted write operations. Only processor bus masters can modify the gate registers. Once locked, a gate can (and must) be opened (unlocked) by the locking processor core.

Multiple gate values can be read in a single access, but only a single gate can be updated via a write operation at a time. 16- and 32-bit writes to multiple gates are allowed, but the write data operand must only update the state of a single gate. A byte write data value of 0x03 is defined as "no operation" and does not affect the state of the corresponding gate register. Attempts to write multiple gates in a single aligned access with a size larger than an 8-bit (byte) reference generate an error termination and do not allow any gate state changes

Address: 0h base + 5h offset = 5h



SEMA4_Gate05 field descriptions

Field	Description
0–5 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
6–7 GTFSM	Gate Finite State Machine. Gate Finite State Machine. The hardware gate is maintained in a 3-state implementation-unlocked, locked by processor 0 or locked by processor 1. For more details, see SEMA4_GATEn Operation . NOTE: The state of the gate reflects the last processor that locked it, which can be useful during system debug. 00 The gate is unlocked (free). 01 The gate has been locked by processor 0. 10 The gate has been locked by processor 1. 11 This state encoding is never used and therefore reserved. Attempted writes of 0x03 are treated as "no operation" and do not affect the gate state machine.

13.3.7 Semaphores Gate 6 Register (SEMA4_Gate06)

Each semaphore gate is implemented in a 2-bit finite state machine, right-justified in a byte data structure. The hardware uses the bus master number in conjunction with the data patterns to validate all attempted write operations. Only processor bus masters can modify the gate registers. Once locked, a gate can (and must) be opened (unlocked) by the locking processor core.

Multiple gate values can be read in a single access, but only a single gate can be updated via a write operation at a time. 16- and 32-bit writes to multiple gates are allowed, but the write data operand must only update the state of a single gate. A byte write data value of 0x03 is defined as "no operation" and does not affect the state of the corresponding gate register. Attempts to write multiple gates in a single aligned access with a size larger than an 8-bit (byte) reference generate an error termination and do not allow any gate state changes

Address: 0h base + 6h offset = 6h

Bit	0	1	2	3	4	5	6	7
Read	0					GTFSM		
Write								
Reset	0	0	0	0	0	0	0	0

SEMA4_Gate06 field descriptions

Field	Description
0–5 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
6–7 GTFSM	Gate Finite State Machine. Gate Finite State Machine. The hardware gate is maintained in a 3-state implementation-unlocked, locked by processor 0 or locked by processor 1. For more details, see SEMA4_GATEn Operation . NOTE: The state of the gate reflects the last processor that locked it, which can be useful during system debug. 00 The gate is unlocked (free). 01 The gate has been locked by processor 0. 10 The gate has been locked by processor 1. 11 This state encoding is never used and therefore reserved. Attempted writes of 0x03 are treated as "no operation" and do not affect the gate state machine.

13.3.8 Semaphores Gate 7 Register (SEMA4_Gate07)

Each semaphore gate is implemented in a 2-bit finite state machine, right-justified in a byte data structure. The hardware uses the bus master number in conjunction with the data patterns to validate all attempted write operations. Only processor bus masters can modify the gate registers. Once locked, a gate can (and must) be opened (unlocked) by the locking processor core.

Multiple gate values can be read in a single access, but only a single gate can be updated via a write operation at a time. 16- and 32-bit writes to multiple gates are allowed, but the write data operand must only update the state of a single gate. A byte write data value of 0x03 is defined as "no operation" and does not affect the state of the corresponding gate register. Attempts to write multiple gates in a single aligned access with a size larger than an 8-bit (byte) reference generate an error termination and do not allow any gate state changes

Address: 0h base + 7h offset = 7h



SEMA4_Gate07 field descriptions

Field	Description
0–5 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
6–7 GTFSM	Gate Finite State Machine. Gate Finite State Machine. The hardware gate is maintained in a 3-state implementation-unlocked, locked by processor 0 or locked by processor 1. For more details, see SEMA4_GATEn Operation . NOTE: The state of the gate reflects the last processor that locked it, which can be useful during system debug. 00 The gate is unlocked (free). 01 The gate has been locked by processor 0. 10 The gate has been locked by processor 1. 11 This state encoding is never used and therefore reserved. Attempted writes of 0x03 are treated as "no operation" and do not affect the gate state machine.

13.3.9 Semaphores Gate 8 Register (SEMA4_Gate08)

Each semaphore gate is implemented in a 2-bit finite state machine, right-justified in a byte data structure. The hardware uses the bus master number in conjunction with the data patterns to validate all attempted write operations. Only processor bus masters can modify the gate registers. Once locked, a gate can (and must) be opened (unlocked) by the locking processor core.

Multiple gate values can be read in a single access, but only a single gate can be updated via a write operation at a time. 16- and 32-bit writes to multiple gates are allowed, but the write data operand must only update the state of a single gate. A byte write data value of 0x03 is defined as "no operation" and does not affect the state of the corresponding gate register. Attempts to write multiple gates in a single aligned access with a size larger than an 8-bit (byte) reference generate an error termination and do not allow any gate state changes

Address: 0h base + 8h offset = 8h

Bit	0	1	2	3	4	5	6	7
Read	0					GTFSM		
Write								
Reset	0	0	0	0	0	0	0	0

SEMA4_Gate08 field descriptions

Field	Description
0–5 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
6–7 GTFSM	Gate Finite State Machine. Gate Finite State Machine. The hardware gate is maintained in a 3-state implementation-unlocked, locked by processor 0 or locked by processor 1. For more details, see SEMA4_GATEn Operation . NOTE: The state of the gate reflects the last processor that locked it, which can be useful during system debug. 00 The gate is unlocked (free). 01 The gate has been locked by processor 0. 10 The gate has been locked by processor 1. 11 This state encoding is never used and therefore reserved. Attempted writes of 0x03 are treated as "no operation" and do not affect the gate state machine.

13.3.10 Semaphores Gate 9 Register (SEMA4_Gate09)

Each semaphore gate is implemented in a 2-bit finite state machine, right-justified in a byte data structure. The hardware uses the bus master number in conjunction with the data patterns to validate all attempted write operations. Only processor bus masters can modify the gate registers. Once locked, a gate can (and must) be opened (unlocked) by the locking processor core.

Multiple gate values can be read in a single access, but only a single gate can be updated via a write operation at a time. 16- and 32-bit writes to multiple gates are allowed, but the write data operand must only update the state of a single gate. A byte write data value of 0x03 is defined as "no operation" and does not affect the state of the corresponding gate register. Attempts to write multiple gates in a single aligned access with a size larger than an 8-bit (byte) reference generate an error termination and do not allow any gate state changes

Address: 0h base + 9h offset = 9h



SEMA4_Gate09 field descriptions

Field	Description
0–5 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
6–7 GTFSM	Gate Finite State Machine. Gate Finite State Machine. The hardware gate is maintained in a 3-state implementation-unlocked, locked by processor 0 or locked by processor 1. For more details, see SEMA4_GATEn Operation . NOTE: The state of the gate reflects the last processor that locked it, which can be useful during system debug. 00 The gate is unlocked (free). 01 The gate has been locked by processor 0. 10 The gate has been locked by processor 1. 11 This state encoding is never used and therefore reserved. Attempted writes of 0x03 are treated as "no operation" and do not affect the gate state machine.

13.3.11 Semaphores Gate 10 Register (SEMA4_Gate10)

Each semaphore gate is implemented in a 2-bit finite state machine, right-justified in a byte data structure. The hardware uses the bus master number in conjunction with the data patterns to validate all attempted write operations. Only processor bus masters can modify the gate registers. Once locked, a gate can (and must) be opened (unlocked) by the locking processor core.

Multiple gate values can be read in a single access, but only a single gate can be updated via a write operation at a time. 16- and 32-bit writes to multiple gates are allowed, but the write data operand must only update the state of a single gate. A byte write data value of 0x03 is defined as "no operation" and does not affect the state of the corresponding gate register. Attempts to write multiple gates in a single aligned access with a size larger than an 8-bit (byte) reference generate an error termination and do not allow any gate state changes

Address: 0h base + Ah offset = Ah

Bit	0	1	2	3	4	5	6	7
Read	0					GTFSM		
Write								
Reset	0	0	0	0	0	0	0	0

SEMA4_Gate10 field descriptions

Field	Description
0–5 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
6–7 GTFSM	Gate Finite State Machine. Gate Finite State Machine. The hardware gate is maintained in a 3-state implementation-unlocked, locked by processor 0 or locked by processor 1. For more details, see SEMA4_GATEn Operation . NOTE: The state of the gate reflects the last processor that locked it, which can be useful during system debug. 00 The gate is unlocked (free). 01 The gate has been locked by processor 0. 10 The gate has been locked by processor 1. 11 This state encoding is never used and therefore reserved. Attempted writes of 0x03 are treated as "no operation" and do not affect the gate state machine.

13.3.12 Semaphores Gate 11 Register (SEMA4_Gate11)

Each semaphore gate is implemented in a 2-bit finite state machine, right-justified in a byte data structure. The hardware uses the bus master number in conjunction with the data patterns to validate all attempted write operations. Only processor bus masters can modify the gate registers. Once locked, a gate can (and must) be opened (unlocked) by the locking processor core.

Multiple gate values can be read in a single access, but only a single gate can be updated via a write operation at a time. 16- and 32-bit writes to multiple gates are allowed, but the write data operand must only update the state of a single gate. A byte write data value of 0x03 is defined as "no operation" and does not affect the state of the corresponding gate register. Attempts to write multiple gates in a single aligned access with a size larger than an 8-bit (byte) reference generate an error termination and do not allow any gate state changes

Address: 0h base + Bh offset = Bh



SEMA4_Gate11 field descriptions

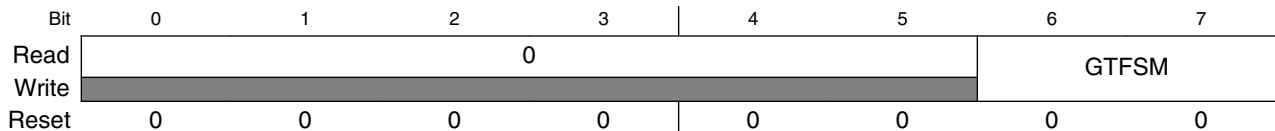
Field	Description
0–5 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
6–7 GTFSM	Gate Finite State Machine. Gate Finite State Machine. The hardware gate is maintained in a 3-state implementation-unlocked, locked by processor 0 or locked by processor 1. For more details, see SEMA4_GATEn Operation . NOTE: The state of the gate reflects the last processor that locked it, which can be useful during system debug. 00 The gate is unlocked (free). 01 The gate has been locked by processor 0. 10 The gate has been locked by processor 1. 11 This state encoding is never used and therefore reserved. Attempted writes of 0x03 are treated as "no operation" and do not affect the gate state machine.

13.3.13 Semaphores Gate 12 Register (SEMA4_Gate12)

Each semaphore gate is implemented in a 2-bit finite state machine, right-justified in a byte data structure. The hardware uses the bus master number in conjunction with the data patterns to validate all attempted write operations. Only processor bus masters can modify the gate registers. Once locked, a gate can (and must) be opened (unlocked) by the locking processor core.

Multiple gate values can be read in a single access, but only a single gate can be updated via a write operation at a time. 16- and 32-bit writes to multiple gates are allowed, but the write data operand must only update the state of a single gate. A byte write data value of 0x03 is defined as "no operation" and does not affect the state of the corresponding gate register. Attempts to write multiple gates in a single aligned access with a size larger than an 8-bit (byte) reference generate an error termination and do not allow any gate state changes

Address: 0h base + Ch offset = Ch



SEMA4_Gate12 field descriptions

Field	Description
0–5 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
6–7 GTFSM	Gate Finite State Machine. Gate Finite State Machine. The hardware gate is maintained in a 3-state implementation-unlocked, locked by processor 0 or locked by processor 1. For more details, see SEMA4_GATEn Operation . NOTE: The state of the gate reflects the last processor that locked it, which can be useful during system debug. 00 The gate is unlocked (free). 01 The gate has been locked by processor 0. 10 The gate has been locked by processor 1. 11 This state encoding is never used and therefore reserved. Attempted writes of 0x03 are treated as "no operation" and do not affect the gate state machine.

13.3.14 Semaphores Gate 13 Register (SEMA4_Gate13)

Each semaphore gate is implemented in a 2-bit finite state machine, right-justified in a byte data structure. The hardware uses the bus master number in conjunction with the data patterns to validate all attempted write operations. Only processor bus masters can modify the gate registers. Once locked, a gate can (and must) be opened (unlocked) by the locking processor core.

Multiple gate values can be read in a single access, but only a single gate can be updated via a write operation at a time. 16- and 32-bit writes to multiple gates are allowed, but the write data operand must only update the state of a single gate. A byte write data value of 0x03 is defined as "no operation" and does not affect the state of the corresponding gate register. Attempts to write multiple gates in a single aligned access with a size larger than an 8-bit (byte) reference generate an error termination and do not allow any gate state changes

Address: 0h base + Dh offset = Dh



SEMA4_Gate13 field descriptions

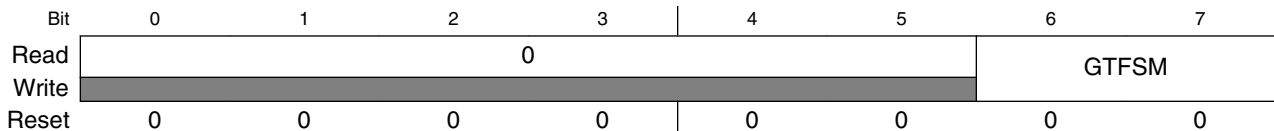
Field	Description
0–5 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
6–7 GTFSM	Gate Finite State Machine. Gate Finite State Machine. The hardware gate is maintained in a 3-state implementation-unlocked, locked by processor 0 or locked by processor 1. For more details, see SEMA4_GATEn Operation . NOTE: The state of the gate reflects the last processor that locked it, which can be useful during system debug. 00 The gate is unlocked (free). 01 The gate has been locked by processor 0. 10 The gate has been locked by processor 1. 11 This state encoding is never used and therefore reserved. Attempted writes of 0x03 are treated as "no operation" and do not affect the gate state machine.

13.3.15 Semaphores Gate 14 Register (SEMA4_Gate14)

Each semaphore gate is implemented in a 2-bit finite state machine, right-justified in a byte data structure. The hardware uses the bus master number in conjunction with the data patterns to validate all attempted write operations. Only processor bus masters can modify the gate registers. Once locked, a gate can (and must) be opened (unlocked) by the locking processor core.

Multiple gate values can be read in a single access, but only a single gate can be updated via a write operation at a time. 16- and 32-bit writes to multiple gates are allowed, but the write data operand must only update the state of a single gate. A byte write data value of 0x03 is defined as "no operation" and does not affect the state of the corresponding gate register. Attempts to write multiple gates in a single aligned access with a size larger than an 8-bit (byte) reference generate an error termination and do not allow any gate state changes

Address: 0h base + Eh offset = Eh



SEMA4_Gate14 field descriptions

Field	Description
0–5 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
6–7 GTFSM	Gate Finite State Machine. Gate Finite State Machine. The hardware gate is maintained in a 3-state implementation-unlocked, locked by processor 0 or locked by processor 1. For more details, see SEMA4_GATEn Operation . NOTE: The state of the gate reflects the last processor that locked it, which can be useful during system debug. 00 The gate is unlocked (free). 01 The gate has been locked by processor 0. 10 The gate has been locked by processor 1. 11 This state encoding is never used and therefore reserved. Attempted writes of 0x03 are treated as "no operation" and do not affect the gate state machine.

13.3.16 Semaphores Gate 15 Register (SEMA4_Gate15)

Each semaphore gate is implemented in a 2-bit finite state machine, right-justified in a byte data structure. The hardware uses the bus master number in conjunction with the data patterns to validate all attempted write operations. Only processor bus masters can modify the gate registers. Once locked, a gate can (and must) be opened (unlocked) by the locking processor core.

Multiple gate values can be read in a single access, but only a single gate can be updated via a write operation at a time. 16- and 32-bit writes to multiple gates are allowed, but the write data operand must only update the state of a single gate. A byte write data value of 0x03 is defined as "no operation" and does not affect the state of the corresponding gate register. Attempts to write multiple gates in a single aligned access with a size larger than an 8-bit (byte) reference generate an error termination and do not allow any gate state changes

Address: 0h base + Fh offset = Fh



SEMA4_Gate15 field descriptions

Field	Description
0–5 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
6–7 GTFSM	Gate Finite State Machine. Gate Finite State Machine. The hardware gate is maintained in a 3-state implementation-unlocked, locked by processor 0 or locked by processor 1. For more details, see SEMA4_GATEn Operation . NOTE: The state of the gate reflects the last processor that locked it, which can be useful during system debug. 00 The gate is unlocked (free). 01 The gate has been locked by processor 0. 10 The gate has been locked by processor 1. 11 This state encoding is never used and therefore reserved. Attempted writes of 0x03 are treated as "no operation" and do not affect the gate state machine.

13.3.17 Semaphores Processor n IRQ Notification Enable (SEMA4_CPnINE)

The application of a hardware semaphore module provides an opportunity for implementation of helpful system-level features. An example is an optional mechanism to generate a processor interrupt after a failed lock attempt. Recall traditional software gate functions execute a spin-wait loop in an effort to obtain and lock the referenced gate. With this module, the processor that fails in the lock attempt could continue with other tasks and allow a properly-enabled notification interrupt to return its execution to the original lock function.

The optional notification interrupt function consists of two registers for each processor: an interrupt notification enable register (SEMA4_CPnINE) and the interrupt request register (SEMA4_CPnNTF). To support implementations with more than 16 gates, these registers can be referenced with aligned 16- or 32-bit accesses. For the SEMA4_CPnINE registers, unimplemented bits read as zeroes, and writes are ignored.

Address: 0h base + 40h offset + (8d × i), where i=0d to 1d

Bit	0	1	2	3	4	5	6	7
Read	INE0	INE1	INE2	INE3	INE4	INE5	INE6	INE7
Write								
Reset	0	0	0	0	0	0	0	0

Bit	8	9	10	11	12	13	14	15
Read	INE8	INE9	INE10	INE11	INE12	INE13	INE14	INE15
Write								
Reset	0	0	0	0	0	0	0	0

SEMA4_CPnINE field descriptions

Field	Description
0 INE0	Interrupt Request Notification Enable 0. This field is a bitmap to enable the generation of an interrupt notification from a failed attempt to lock gate 0. 0 The generation of the notification interrupt is disabled. 1 The generation of the notification interrupt is enabled.
1 INE1	Interrupt Request Notification Enable 1. This field is a bitmap to enable the generation of an interrupt notification from a failed attempt to lock gate 1. 0 The generation of the notification interrupt is disabled. 1 The generation of the notification interrupt is enabled.
2 INE2	Interrupt Request Notification Enable 2. This field is a bitmap to enable the generation of an interrupt notification from a failed attempt to lock gate 2. 0 The generation of the notification interrupt is disabled. 1 The generation of the notification interrupt is enabled.
3 INE3	Interrupt Request Notification Enable 3. This field is a bitmap to enable the generation of an interrupt notification from a failed attempt to lock gate 3.

Table continues on the next page...

SEMA4_CPnINE field descriptions (continued)

Field	Description
	0 The generation of the notification interrupt is disabled. 1 The generation of the notification interrupt is enabled.
4 INE4	Interrupt Request Notification Enable 4. This field is a bitmap to enable the generation of an interrupt notification from a failed attempt to lock gate 4. 0 The generation of the notification interrupt is disabled. 1 The generation of the notification interrupt is enabled.
5 INE5	Interrupt Request Notification Enable 5. This field is a bitmap to enable the generation of an interrupt notification from a failed attempt to lock gate 5. 0 The generation of the notification interrupt is disabled. 1 The generation of the notification interrupt is enabled.
6 INE6	Interrupt Request Notification Enable 6. This field is a bitmap to enable the generation of an interrupt notification from a failed attempt to lock gate 6. 0 The generation of the notification interrupt is disabled. 1 The generation of the notification interrupt is enabled.
7 INE7	Interrupt Request Notification Enable 7. This field is a bitmap to enable the generation of an interrupt notification from a failed attempt to lock gate 7. 0 The generation of the notification interrupt is disabled. 1 The generation of the notification interrupt is enabled.
8 INE8	Interrupt Request Notification Enable 8. This field is a bitmap to enable the generation of an interrupt notification from a failed attempt to lock gate 8. 0 The generation of the notification interrupt is disabled. 1 The generation of the notification interrupt is enabled.
9 INE9	Interrupt Request Notification Enable 9. This field is a bitmap to enable the generation of an interrupt notification from a failed attempt to lock gate 9. 0 The generation of the notification interrupt is disabled. 1 The generation of the notification interrupt is enabled.
10 INE10	Interrupt Request Notification Enable 10. This field is a bitmap to enable the generation of an interrupt notification from a failed attempt to lock gate 10. 0 The generation of the notification interrupt is disabled. 1 The generation of the notification interrupt is enabled.
11 INE11	Interrupt Request Notification Enable 11. This field is a bitmap to enable the generation of an interrupt notification from a failed attempt to lock gate 11. 0 The generation of the notification interrupt is disabled. 1 The generation of the notification interrupt is enabled.
12 INE12	Interrupt Request Notification Enable 12. This field is a bitmap to enable the generation of an interrupt notification from a failed attempt to lock gate 12. 0 The generation of the notification interrupt is disabled. 1 The generation of the notification interrupt is enabled.
13 INE13	Interrupt Request Notification Enable 13. This field is a bitmap to enable the generation of an interrupt notification from a failed attempt to lock gate 13.

Table continues on the next page...

SEMA4_CPnINE field descriptions (continued)

Field	Description
	0 The generation of the notification interrupt is disabled. 1 The generation of the notification interrupt is enabled.
14 INE14	Interrupt Request Notification Enable 14. This field is a bitmap to enable the generation of an interrupt notification from a failed attempt to lock gate 14. 0 The generation of the notification interrupt is disabled. 1 The generation of the notification interrupt is enabled.
15 INE15	Interrupt Request Notification Enable 15. This field is a bitmap to enable the generation of an interrupt notification from a failed attempt to lock gate 15. 0 The generation of the notification interrupt is disabled. 1 The generation of the notification interrupt is enabled.

13.3.18 Semaphores Processor n IRQ Notification (SEMA4_CPnNTF)

The Semaphores module optionally allows the processor that fails in the lock attempt to continue with other tasks and allow a properly-enabled notification interrupt to return its execution to the original lock function rather than simply execute in a spin-wait loop.

The optional notification interrupt mechanism consists of two registers for each processor: an interrupt notification enable register (SEMA4_CPnINE) and the read-only notification interrupt request register (SEMA4_CPnNTF). To support implementations with more than 16 gates, these registers can be referenced with aligned 16- or 32-bit accesses. For the SEMA4_CPnNTF registers, unimplemented bits read as zeroes.

The notification interrupt is generated via a unique finite state machine, one per hardware gate. This machine operates in the following manner:

1. When an attempted lock fails, the FSM enters a first state where it waits until the gate is unlocked.
2. Once unlocked, the FSM enters a second state where it generates an interrupt request to the “failed lock” processor.
3. When the “failed lock” processor succeeds in locking the gate, the IRQ is automatically negated and the FSM returns to the idle state. However, if the other processor again locks the gate, the FSM returns to the first state, negates the interrupt request, and then waits for the gate to be unlocked (again).

The notification interrupt request is implemented in a 3-bit, 5-state machine, where two specific states are encoded and program-visible as SEMA4_CP0NTF[GNn] and SEMA4_CP1NTF[GNn].

Memory map and register definition

Address: 0h base + 80h offset + (8d × i), where i=0d to 1d

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Read	GNn															
Write																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

SEMA4_CPnNTF field descriptions

Field	Description
0–15 GNn	Gate n Notification. This read-only field is a bitmap of the interrupt request notification from a failed attempt to lock gate n. For more details, see SEMA4_CPnNTF Operation .

13.3.19 Semaphores (Secure) Reset Gate n (SEMA4_RSTGT)

Although the intent of the hardware gate implementation specifies a protocol where the locking processor must unlock the gate, it is recognized that system operation may require a reset function to re-initialize the state of any gate(s) without requiring a system-level reset.

To support this special gate reset requirement, the Semaphores module implements a "secure" reset mechanism which allows a hardware gate (or all the gates) to be initialized by following a specific dual-write access pattern. Using a technique similar to that required for the servicing of a software watchdog timer, the secure gate reset requires two consecutive writes with predefined data patterns from the same processor to force the clearing of the specified gate(s). The required access pattern is:

1. A processor performs a 16-bit write to the SEMA4_RSTGT memory location. The most significant byte (SEMA4_RSTGT[RSTGDP]) must be 0xE2; the least significant byte is a "don't_care" for this reference.
2. The same processor then performs a second 16-bit write to the SEMA4_RSTGT location. For this write, the upper byte (SEMA4_RSTGT[RSTGDP]) is the logical complement of the first data pattern (0x1D) and the lower byte (SEMA4_RSTGT[RSTGTN]) specifies the gate(s) to be reset. This gate field can specify a single gate be cleared, or that all gates are cleared.
3. Reads of the SEMA4_RSTGT location return information on the 2-bit state machine (SEMA4_RSTGT[RSTGSM]) which implements this function, the bus master performing the reset (SEMA4_RSTGT[RSTGMS]) and the gate number(s) last cleared (SEMA4_RSTGT[RSTGTN]). Reads of the SEMA4_RSTGT register do not affect the secure reset finite state machine in any manner.

Address: 0h base + 100h offset = 100h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Read	RSTGSM_RSTGMS_RSTGDP								RSTGTN							
Write																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

SEMA4_RSTGT field descriptions

Field	Description																							
0–7 RSTGSM_ RSTGMS_ RSTGDP	NOTE: This field contains subfields that vary depending on whether it is being read or written. Sub-fields indicated as having read access are valid only for read operations. Sub-fields indicated as having write access are valid only for write operations. Bit numbering in the descriptions begins with the most significant bit numbered 0. See the following table for details. <table><tr><th>Access</th><th>Sub-Field</th><th>Description</th></tr><tr><td rowspan="7">Read-Only</td><td>0-1 Reserved</td><td>Reserved. Always reads 0.</td></tr><tr><td rowspan="4">2-3 RSTGSM</td><td>Reset Gate Finite State Machine. Reads of the SEMA4_RSTGT register return the encoded state machine value. The reset state machine is maintained in a 2-bit, 3-state implementation, defined as:</td></tr><tr><td>00</td><td>Idle, waiting for the first data pattern write.</td></tr><tr><td>01</td><td>Waiting for the second data pattern write.</td></tr><tr><td>10</td><td>The 2-write sequence has completed. Generate the specified gate reset(s). After the reset is performed, this machine returns to the idle (waiting for first data pattern write) state. Note that the RSTGSM = 0b10 state is valid for only a single machine cycle, so it is impossible for a read to return this value</td></tr><tr><td>11</td><td>This state encoding is never used and therefore reserved.</td></tr><tr><td>4 Reserved</td><td>Reserved. Always reads 0.</td></tr><tr><td>5-7 RSTGMS</td><td>Reset Gate Bus Master. This 3-bit read-only field records the logical number of the bus master performing the gate reset function. The reset function requires that the two consecutive writes to this register be initiated by the same bus master to succeed. This field is updated each time a write to this register occurs. The association between system bus master port numbers, the associated bus master device and the logical processor number is SoC-specific. See the chip configuration chapter for this information.</td></tr><tr><td>Write-Only</td><td>0-7 RSTGDP</td><td>Reset Gate Data Pattern. This write-only field is accessed with the specified data patterns on the two consecutive writes to enable the gate reset mechanism. For the first write, RSTGDP = 0xE2 while the second write requires RSTGDP = 0x1D.</td></tr></table>	Access	Sub-Field	Description	Read-Only	0-1 Reserved	Reserved. Always reads 0.	2-3 RSTGSM	Reset Gate Finite State Machine. Reads of the SEMA4_RSTGT register return the encoded state machine value. The reset state machine is maintained in a 2-bit, 3-state implementation, defined as:	00	Idle, waiting for the first data pattern write.	01	Waiting for the second data pattern write.	10	The 2-write sequence has completed. Generate the specified gate reset(s). After the reset is performed, this machine returns to the idle (waiting for first data pattern write) state. Note that the RSTGSM = 0b10 state is valid for only a single machine cycle, so it is impossible for a read to return this value	11	This state encoding is never used and therefore reserved.	4 Reserved	Reserved. Always reads 0.	5-7 RSTGMS	Reset Gate Bus Master. This 3-bit read-only field records the logical number of the bus master performing the gate reset function. The reset function requires that the two consecutive writes to this register be initiated by the same bus master to succeed. This field is updated each time a write to this register occurs. The association between system bus master port numbers, the associated bus master device and the logical processor number is SoC-specific. See the chip configuration chapter for this information.	Write-Only	0-7 RSTGDP	Reset Gate Data Pattern. This write-only field is accessed with the specified data patterns on the two consecutive writes to enable the gate reset mechanism. For the first write, RSTGDP = 0xE2 while the second write requires RSTGDP = 0x1D.
Access	Sub-Field	Description																						
Read-Only	0-1 Reserved	Reserved. Always reads 0.																						
	2-3 RSTGSM	Reset Gate Finite State Machine. Reads of the SEMA4_RSTGT register return the encoded state machine value. The reset state machine is maintained in a 2-bit, 3-state implementation, defined as:																						
		00	Idle, waiting for the first data pattern write.																					
		01	Waiting for the second data pattern write.																					
		10	The 2-write sequence has completed. Generate the specified gate reset(s). After the reset is performed, this machine returns to the idle (waiting for first data pattern write) state. Note that the RSTGSM = 0b10 state is valid for only a single machine cycle, so it is impossible for a read to return this value																					
	11	This state encoding is never used and therefore reserved.																						
	4 Reserved	Reserved. Always reads 0.																						
5-7 RSTGMS	Reset Gate Bus Master. This 3-bit read-only field records the logical number of the bus master performing the gate reset function. The reset function requires that the two consecutive writes to this register be initiated by the same bus master to succeed. This field is updated each time a write to this register occurs. The association between system bus master port numbers, the associated bus master device and the logical processor number is SoC-specific. See the chip configuration chapter for this information.																							
Write-Only	0-7 RSTGDP	Reset Gate Data Pattern. This write-only field is accessed with the specified data patterns on the two consecutive writes to enable the gate reset mechanism. For the first write, RSTGDP = 0xE2 while the second write requires RSTGDP = 0x1D.																						
8–15 RSTGTN	Reset Gate Number. This 8-bit field specifies the specific hardware gate to be reset. This field is updated by the second write. If RSTGTN < 64, then reset the single gate defined by RSTGTN, else reset all the gates. The corresponding secure IRQ notification state machine(s) are also reset.																							

13.3.20 Semaphores (Secure) Reset IRQ Notification (SEMA4_RSTNTF)

As with the case of the secure reset function and the hardware gates, it is recognized that system operation may require a reset function to re-initialize the state of the IRQ notification logic without requiring a system-level reset.

To support this special notification reset requirement, the Semaphores module implements a "secure" reset mechanism which allows an IRQ notification (or all the notifications) to be initialized by following a specific dual-write access pattern. When successful, the specified IRQ notification state machine(s) are reset. Using a technique similar to that required for the servicing of a software watchdog timer, the secure reset mechanism requires two consecutive writes with predefined data patterns from the same processor to force the clearing of the IRQ notification(s). The required access pattern is:

- 1. A processor performs a 16-bit write to the SEMA4_RSTNTF memory location. The most significant byte (SEMA4_RSTNTF[RSTNDP]) must be 0x47; the least significant byte is a "don't_care" for this reference.
- 2. The same processor then performs a second 16-bit write to the SEMA4_RSTNTF location. For this write, the upper byte (SEMA4_RSTNTF[RSTNDP]) is the logical complement of the first data pattern (0xB8) and the lower byte (SEMA4_RSTNTF[RSTNTN]) specifies the notification(s) to be reset. This field can specify a single notification be cleared, or that all notifications are cleared.
- 3. Reads of the SEMA4_RSTNTF location return information on the 2-bit state machine (SEMA4_RSTNTF[RSTNSM]) which implements this function, the bus master performing the reset (SEMA4_RSTNTF[RSTNMS]) and the notification number(s) last cleared (SEMA4_RSTNTF[RSTNTN]). Reads of the SEMA4_RSTNTF register do not affect the secure reset finite state machine in any manner.

Address: 0h base + 104h offset = 104h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Read	RSTNSM_RSTNMS_RSTNDP								RSTNTN							
Write																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

SEMA4_RSTNTF field descriptions

Field	Description
0-7 RSTNSM_ RSTNMS_ RSTNDP	NOTE: This field contains subfields that vary depending on whether it is being read or written. Sub-fields indicated as having read access are valid only for read operations. Sub-fields indicated as having write access are valid only for write operations. Bit numbering in the descriptions begins with the most significant bit numbered 0. See the following table for details.

Table continues on the next page...

SEMA4_RSTNTF field descriptions (continued)

Field	Description		
	Access	Sub-Field	Description
	Read-Only	0-1 Reserved	Reserved. Always reads 0.
		2-3 RSTNSM	Reset Notification Finite State Machine. Reads of the SEMA4_RSTNTF register return the encoded state machine value. The reset state machine is maintained in a 2-bit, 3-state implementation, defined as:
			00 Idle, waiting for the first data pattern write.
			01 Waiting for the second data pattern write.
			10 The 2-write sequence has completed. Generate the specified notification reset(s). After the reset is performed, this machine returns to the idle (waiting for first data pattern write) state. Note the RSTNSM = 10 state is valid for only a single machine cycle, so it is impossible for a read to return this value.
			11 This state encoding is never used and therefore reserved..
		4 Reserved	Reserved. Always reads 0.
		5-7 RSTNMS	Reset Notification Bus Master. This 3-bit read-only field records the logical number of the bus master performing the notification reset function. The reset function requires that the two consecutive writes to this register be initiated by the same bus master to succeed. This field is updated each time a write to this register occurs. The association between system bus master port numbers, the associated bus master device and the logical processor number is SoC-specific. See the chip configuration chapter for this information.
	Write-Only	0-7 RSTNDP	Reset Notification Data Pattern. This write-only field is accessed with the specified data patterns on the two consecutive writes to enable the notification reset mechanism. For the first write, RSTNDP = 0x47 while the second write requires RSTNDP = 0xb8.
8-15 RSTNTN	Reset Notification Number. This 8-bit field specifies the specific IRQ notification state machine to be reset. This field is updated by the second write. If RSTNTN < 64, then reset the single IRQ notification machine defined by RSTNTN, else reset all the notifications.		

13.4 Functional Description

In this section, the functional operation of the Semaphores module, specifically the state machines of the SEMA4_GATn and SEMA4_CPnNTF registers are detailed.

13.4.1 SEMA4_GATEn Operation

Recall each of the SEMA4_GATEn registers implements a 2-bit, 3-state machine. The state transitions for each gate are shown in the following figure.

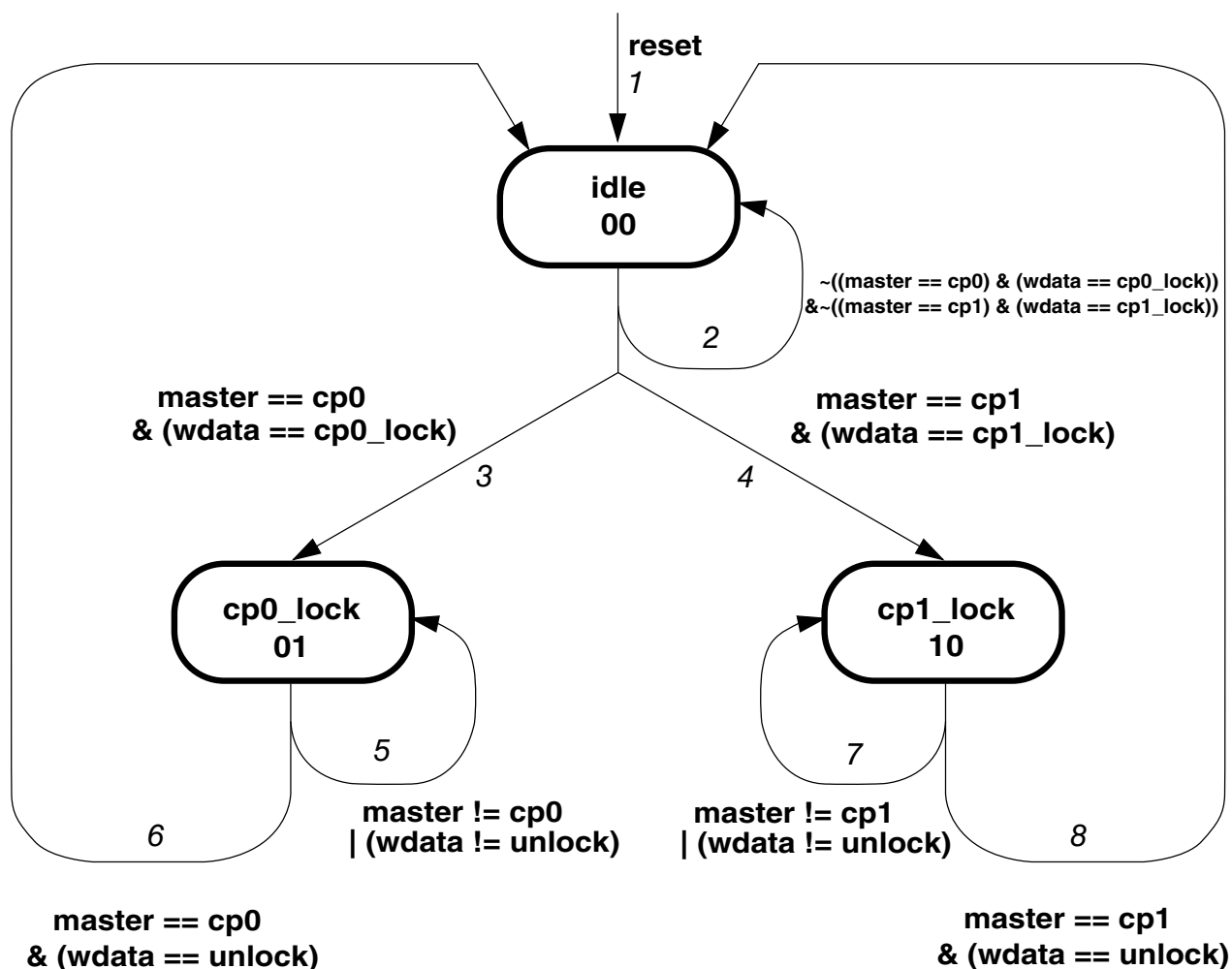


Figure 13-26. SEMA4_GATEn State Machine

The bus master number is used to identify core processor 0 (cp0) or core processor 1 (cp1). The Standard (or Reduced) Product Platform passes the AHB bus master number (`hmaster[2:0]`) through the AIPS (or AIPS-Lite) controller and drives an `aips_master[2:0]` output to the Semaphores module as an IPS sideband signal.

The state transitions for SEMA4_GATEn are defined in the following table.

Table 13-26. SEMA4_GATEn State Transitions

Current State	Next State	Transition	Description
—	idle	1	Any reset, whether a system reset or an individual gate reset, unconditionally forces the gate into the idle state.

Table continues on the next page...

**Table 13-26. SEMA4_GATEn State Transitions
(continued)**

Current State	Next State	Transition	Description
idle	idle	2	Unless a write of the appropriate lock value from the corresponding processor occurs, the gate remains in the idle state.
idle	cp0_lock	3	When a write of the "cp0_lock" data value is initiated by processor 0, the gate transitions into the cp0_lock state.
idle	cp1_lock	4	When a write of the "cp1_lock" value is initiated by processor 1, the gate transitions into the cp1_lock state.
cp0_lock	cp0_lock	5	Once in this state, the gate remains here if any attempted write is not from cp0 with the unlock data value.
cp0_lock	idle	6	The gate returns to the idle (unlocked) state once a write from cp0 with the unlock data value occurs.
cp1_lock	cp1_lock	7	Once in this state, the gate remains here if any attempted write is not from cp1 with the unlock data value.
cp1_lock	idle	8	The gate returns to the idle (unlocked) state once a write from cp1 with the unlock data value occurs.

13.4.2 SEMA4_CPnNTF Operation

The failed lock write notification interrupt request is implemented in a 3-bit, 5-state machine which records failed lock attempts and transitions based on gate locking and unlocking. Two specific states are encoded and program-visible as SEMA4_CP0NTF[GNn] and SEMA4_CP1NTF[GNn]. See the following figure.

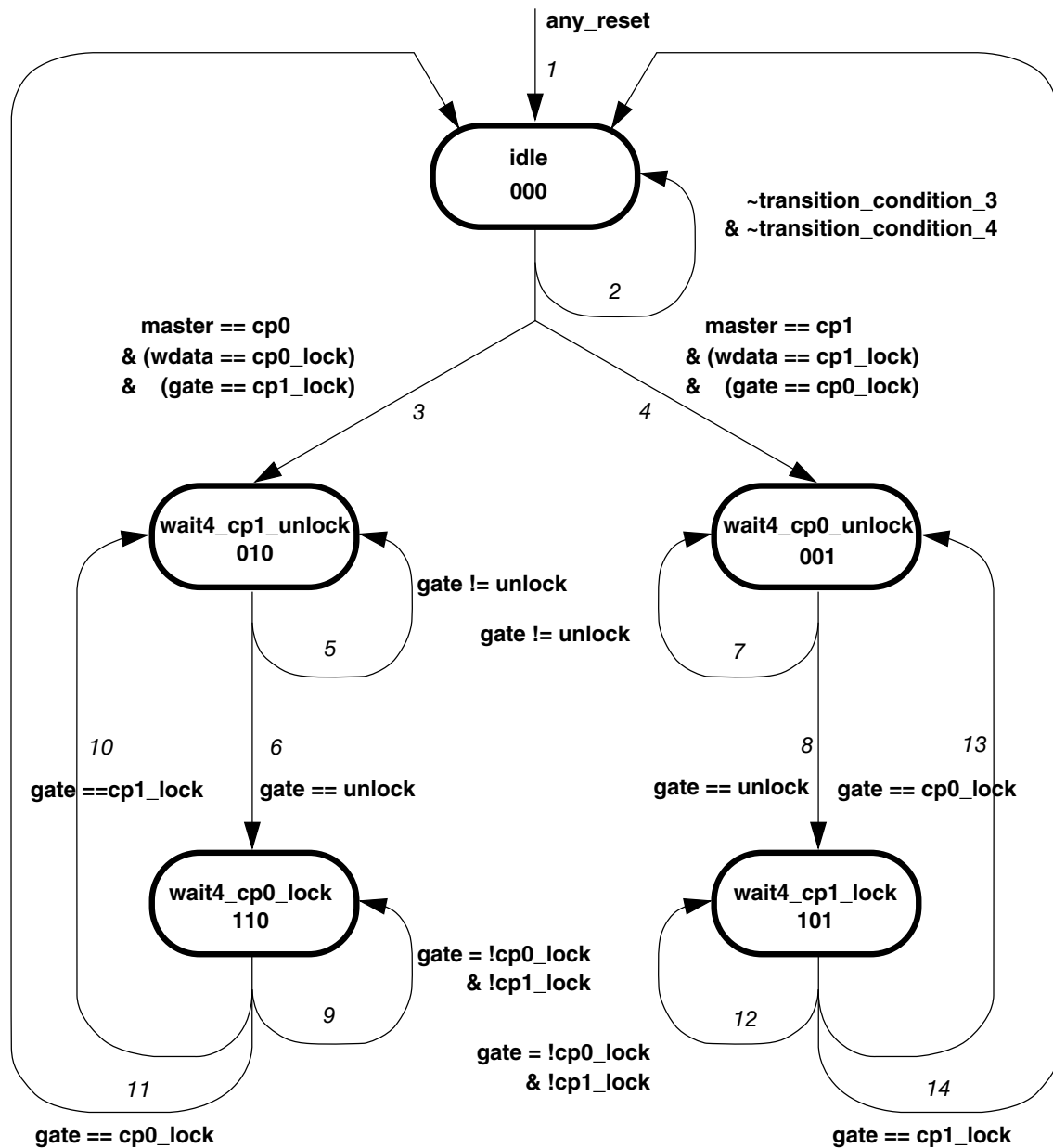


Figure 13-27. IRQ Notification State Machine

The state transitions of the IRQ notification function are defined in the following. Specific states of this machine are program-visible as the SEMA4_CPnNTF registers. In particular, two states are program-visible:

```

if state = wait4_cp0_lock (0b110) // generate cp0_semaphore_int if properly enabled
    then SEMA4_CP0NTF[GNn] = 1; else SEMA4_CP0NTF[GNn] = 0
if state = wait4_cp1_lock (0b101) // generate cp1_semaphore_int if properly enabled
    then SEMA4_CP1NTF[GNn] = 1; else SEMA4_CP1NTF[GNn] = 0
  
```

Table 13-27. IRQ Notification State Transitions

Current State	Next State	Transition	Description
—	idle	1	Any reset, including a system reset or an individual notification or secure gate reset, unconditionally forces the machine into the idle state.
idle	idle	2	Unless a write of the appropriate lock value from the corresponding processor to an already-locked gate occurs, the machine remains in the idle state.
idle	wait4_cp1_unlock	3	When a write of the "cp0_lock" data value is initiated by processor 0 but the gate is already locked by cp1, the machine transitions into this state, where it waits for cp1 to unlock the gate.
idle	wait4_cp0_unlock	4	When a write of the "cp1_lock" data value is initiated by processor 1 but the gate is already locked by cp0, the machine transitions into this state, where it waits for cp0 to unlock the gate.
wait4_cp1_unlock	wait4_cp1_unlock	5	Once in this state, the machine remains here until the gate is unlocked.
wait4_cp1_unlock	wait4_cp0_lock	6	From this state, the machine transitions into the next state, waiting for cp0 to lock the gate, once it has been unlocked.
wait4_cp0_unlock	wait4_cp0_unlock	7	Once in this state, the machine remains here until the gate is unlocked.
wait4_cp0_unlock	wait4_cp1_lock	8	From this state, the machine transitions into the next state, waiting for cp1 to lock the gate, once it has been unlocked.
wait4_cp0_lock	wait4_cp0_lock	9	In this state, the machine generates the notification interrupt (if properly-enabled) and remains here until the gate is locked by processor 0 or the gate is again locked by processor 1.
wait4_cp0_lock	wait4_cp1_unlock	10	In this state, the machine generates the notification interrupt (if properly-enabled) and transitions if the gate is again locked by processor 1. With this transition, the notification interrupt request is negated.
wait4_cp0_lock	idle	11	In this state, the machine generates the notification interrupt (if properly-enabled) and transitions if the gate is finally locked by processor 0. With this transition, the notification interrupt request is negated.
wait4_cp1_lock	wait4_cp1_lock	12	In this state, the machine generates the notification interrupt (if properly-enabled) and remains here until the gate is locked by processor 1 or the gate is again locked by processor 0.
wait4_cp1_lock	wait4_cp0_unlock	13	In this state, the machine generates the notification interrupt (if properly-enabled) and transitions if the gate is again locked by processor 0. With this transition, the notification interrupt request is negated.
wait4_cp1_lock	idle	14	In this state, the machine generates the notification interrupt (if properly-enabled) and transitions if the gate is finally locked by processor 1. With this transition, the notification interrupt request is negated.

The Semaphores module generates two interrupt request output signals, one per processor, combining the SEMA4_CPnINE and SEMA4_CPnNTF registers, where the boolean equations are:

```

cp0_semaphore_int
=   sema4_cp0ine[ine0]    &   sema4_cp0ntf[gn0]
  |   sema4_cp0ine[ine1]    &   sema4_cp0ntf[gn1]
  |   sema4_cp0ine[ine2]    &   sema4_cp0ntf[gn2]
  |   ...
  |   sema4_cp0ine[ine15]   &   sema4_cp0ntf[gn15]
cp1_semaphore_int
=   sema4_cp1ine[ine0]    &   sema4_cp1ntf[gn0]
  |   sema4_cp1ine[ine1]    &   sema4_cp1ntf[gn1]
  |   sema4_cp1ine[ine2]    &   sema4_cp1ntf[gn2]
  |   ...
  |   sema4_cp1ine[ine15]   &   sema4_cp1ntf[gn15]

```

13.5 Initialization Information

The reset state of the IPS_Semaphores module allows it to begin operation without the need for any further initialization. All the internal state machines are cleared by any reset event, allowing the module to immediately begin operation.

13.6 Application Information

In an operational multi-core system, most interactions involving the Semaphores module involves reads and writes to the SEMA4_GATEn registers for implementation of the hardware-enforced software gate functions. Typical code segments for gate functions perform the following operations:

- To lock (close) a gate
 - The processor performs a byte write of "logical_processor_number + 1" to gate[i]
 - The processor reads back gate[i] and checks for a value of "logical_processor_number + 1"

If the compare indicates the expected value, then the gate is locked; proceed with the protected code segment. If the compare does not indicate the expected value, the lock operation failed; repeat the process beginning with byte write to gate[i] in spin-wait loop, or proceed with another execution path and wait for failed lock interrupt notification.

A simple C-language example of a `gateLock` function is shown in the following figure. This function follows the Hennessy/Patterson example described in [Multi-Core Programming 101: Software Gates](#).

```

#define UNLOCK      0
#define CP0_LOCK    1
#define CP2_LOCK    2

void gateLock (n)
int    n;                                /* gate number to lock */
{
    int i;
    int current_value;
    int locked_value;

    i = processor_number(); /* obtain logical CPU number */

    if (i == 0)
        locked_value = CP0_LOCK;
    else
        locked_value = CP1_LOCK;

    /* read the current value of the gate and wait until the state == UNLOCK */
    do {
        current_value = gate[n];
    } while (current_value != UNLOCK);

    /* the current value of the gate == UNLOCK. attempt to lock the gate for this
       processor. spin-wait in this loop until gate ownership is obtained */
    do {
        gate[n] = locked_value; /* write gate with processor_number + 1 */
        current_value = gate[n]; /* read gate to verify ownership was obtained */
    } while (current_value != locked_value);
}

```

Figure 13-28. Sample gateLock Function

- To unlock (open) a gate
 - After completing the protected code segment, the locking processor performs a byte write of zeroes to gate[i], opening (unlocking) the gate

A few comments on the logical CPU number are appropriate. In this example, a reference to `processor_number()` is used to retrieve this hardware configuration value. Typically, the logical processor numbers are defined by a hardwired input vector to the individual cores. The exact method for accessing the logical processor number varies by architecture. For PowerPC cores, there is a processor ID register (PIR) which is SPR 286 and contains this value. A single instruction can be used to move the contents of the PIR into a general-purpose register: `mf spr rx, 286` where `rx` is the destination GPRn. Other architectures may support a specific instruction to move the contents of the logical processor number into a general-purpose register, e.g., `rdcpn rx` for a "read CPU number" instruction.

If the optional failed lock IRQ notification mechanisms are used, then accesses to the related registers (SEMA4_CPnINE, SEMA4_CPnNTF) are required. Note that there is no required negation of the failed lock write notification interrupt as the request is automatically negated by the Semaphores module once the gate has been successfully locked by the "failing" processor.

Finally, in the event a system state requires a software-controlled reset of a gate or IRQ notification register(s), accesses to the secure reset control registers (SEMA4_RSTGT, SEMA4_RSTNTF) are required. For these situations, it is recommended that the appropriate IRQ notification enable(s) (SEMA4_CPnINE) bits be disabled *before* initiating the secure reset 2-write sequence to avoid any race conditions involving spurious notification interrupt requests.

Chapter 14

Memory Protection Unit (MPU)

14.1 Chip-specific MPU information

This chip has two instances of the MPU module. Some feature details vary between the instances.

The following table summarizes the feature differences. The table does not list feature details that the instances share.

Table 14-1. MPU instance feature differences

Feature	MPU_0	MPU_1
Number of slave ports supported	Three numbered 0, 1, 2	Two numbered 1, 2

14.1.1 MPU master and slave assignments

The MPU identifies each XBAR master in terms of the master's logical ID. See the "Logical master ID" assignments in [Table 9-1](#).

Each MPU slave port mediates between an XBAR physical slave port and the module that is assigned to that physical slave port. For each MPU instance, the following tables show the alignment among the MPU slave ports, the XBAR physical slave ports, and the corresponding assigned modules.

Table 14-2. MPU_0 slave ports

MPU_0 slave port	XBAR physical slave port	Module assigned to XBAR physical slave port
0	1	EBI
1	2	PRAMC_0
2	6	AIPS_1 (Peripheral Bridge A)

Table 14-3. MPU_1 slave ports

MPU_1 slave port	XBAR physical slave port	Module assigned to XBAR physical slave port
1	4	PRAMC_1
2	7	AIPS_0 (Peripheral Bridge B)

14.1.2 MPU register reset values

The following table lists MPU register reset values that are specific to this chip. For the reset values of all other MPU registers, see [Memory map/register definition](#).

Table 14-4. Chip-specific MPU register reset values

Register	MPU_0 and MPU_1 reset value	Notes
CESR	0081_3200h	NSP field resets to 0011b ¹

1. CESR[NSP]'s reset value on MPU_1 is 3 even though MPU_1 has only two slave ports.

14.2 Introduction

The memory protection unit (MPU) provides hardware access control for all memory references generated in the device.

14.3 Overview

The MPU concurrently monitors all system bus transactions and evaluates their appropriateness using pre-programmed region descriptors that define memory spaces and their access rights. Memory references that have sufficient access control rights are allowed to complete, while references that are not mapped to any region descriptor or have insufficient rights are terminated with a protection error response.

14.3.1 Block diagram

A simplified block diagram of the MPU module is shown in the following figure.

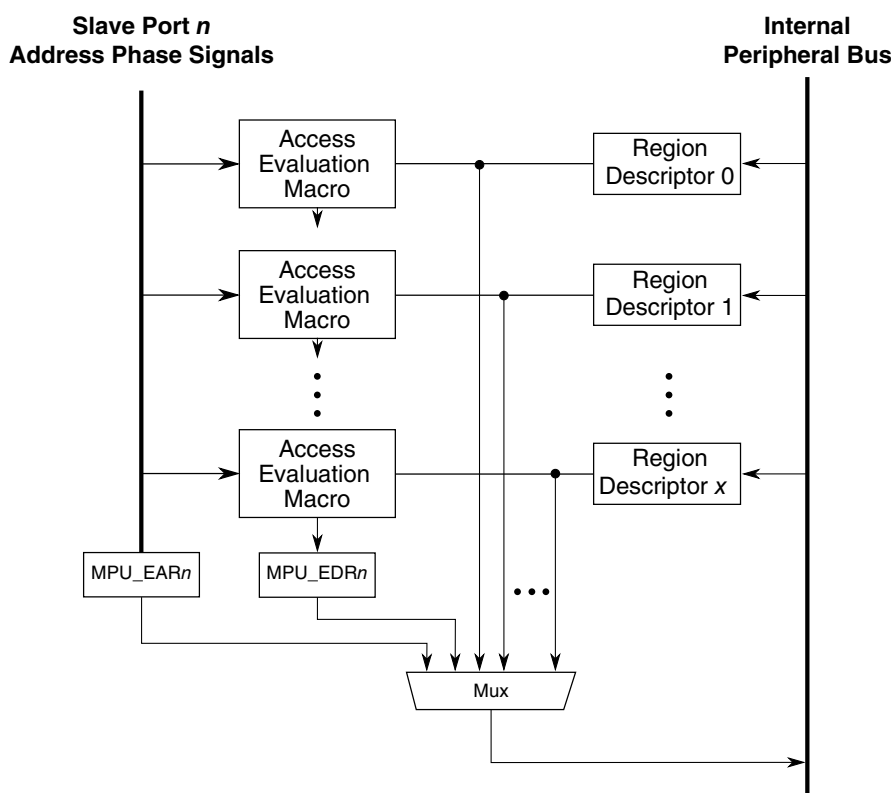


Figure 14-1. MPU block diagram

The hardware's two-dimensional connection matrix is clearly visible with the basic access evaluation macro shown as the replicated submodule block. The crossbar switch slave ports are shown on the left, the region descriptor registers in the middle, and the peripheral bus interface on the right side. The evaluation macro contains two magnitude comparators connected to the start and end address registers from each region descriptor as well as the combinational logic blocks to determine the region hit and the access protection error. For details of the access evaluation macro, see [Access evaluation macro](#).

14.3.2 Features

The MPU implements a two-dimensional hardware array of memory region descriptors and the crossbar slave ports to continuously monitor the legality of every memory reference generated by each bus master in the system.

The feature set includes:

- 16 program-visible 128-bit region descriptors, accessible by four 32-bit words each
 - Each region descriptor defines a modulo-32 byte space, aligned anywhere in memory

- Region sizes can vary from 32 bytes to 4 Gbytes
- Two access control permissions defined in a single descriptor word
 - Masters 0–3: read, write, and execute attributes for supervisor and user accesses
 - Masters 4–7: read and write attributes
- Hardware-assisted maintenance of the descriptor valid bit minimizes coherency issues
- Alternate programming model view of the access control permissions word
- Priority given to granting permission over denying access for overlapping region descriptors
- Detects access protection errors if a memory reference does not hit in any memory region, or if the reference is illegal in all hit memory regions. If an access error occurs, the reference is terminated with an error response, and the MPU inhibits the bus cycle being sent to the targeted slave device.
- Error registers, per slave port, capture the last faulting address, attributes, and other information
- Global MPU enable/disable control bit

14.4 Memory map/register definition

The programming model is partitioned into three groups:

- Control/status registers
- The data structure containing the region descriptors
- The alternate view of the region descriptor access control values

The programming model can be referenced using only 32-bit accesses.

Attempted references of the following types generate an error termination:

- Non-32-bit references
- References to undefined—that is, reserved—addresses
- References with a non-supported access type, such as a write access to a read-only register or a read access of a write-only register

NOTE

See the chip configuration details for any chip-specific register information in this module.

MPU memory map

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/page
0	Control/Error Status Register (MPU_CESR)	32	R/W	See section	14.4.1/540
10	Error Address Register, slave port n (MPU_EAR0)	32	R	0000_0000h	14.4.2/541
14	Error Detail Register, slave port n (MPU_EDR0)	32	R	0000_0000h	14.4.3/542
18	Error Address Register, slave port n (MPU_EAR1)	32	R	0000_0000h	14.4.2/541
1C	Error Detail Register, slave port n (MPU_EDR1)	32	R	0000_0000h	14.4.3/542
20	Error Address Register, slave port n (MPU_EAR2)	32	R	0000_0000h	14.4.2/541
24	Error Detail Register, slave port n (MPU_EDR2)	32	R	0000_0000h	14.4.3/542
400	Region Descriptor n, Word 0 (MPU_RGD0_WORD0)	32	R/W	0000_0000h	14.4.4/543
404	Region Descriptor n, Word 1 (MPU_RGD0_WORD1)	32	R/W	See section	14.4.5/544
408	Region Descriptor n, Word 2 (MPU_RGD0_WORD2)	32	R/W	See section	14.4.6/544
40C	Region Descriptor n, Word 3 (MPU_RGD0_WORD3)	32	R/W	0000_0000h	14.4.7/547
410	Region Descriptor n, Word 0 (MPU_RGD1_WORD0)	32	R/W	0000_0000h	14.4.4/543
414	Region Descriptor n, Word 1 (MPU_RGD1_WORD1)	32	R/W	See section	14.4.5/544
418	Region Descriptor n, Word 2 (MPU_RGD1_WORD2)	32	R/W	See section	14.4.6/544
41C	Region Descriptor n, Word 3 (MPU_RGD1_WORD3)	32	R/W	0000_0000h	14.4.7/547
420	Region Descriptor n, Word 0 (MPU_RGD2_WORD0)	32	R/W	0000_0000h	14.4.4/543
424	Region Descriptor n, Word 1 (MPU_RGD2_WORD1)	32	R/W	See section	14.4.5/544
428	Region Descriptor n, Word 2 (MPU_RGD2_WORD2)	32	R/W	See section	14.4.6/544
42C	Region Descriptor n, Word 3 (MPU_RGD2_WORD3)	32	R/W	0000_0000h	14.4.7/547
430	Region Descriptor n, Word 0 (MPU_RGD3_WORD0)	32	R/W	0000_0000h	14.4.4/543
434	Region Descriptor n, Word 1 (MPU_RGD3_WORD1)	32	R/W	See section	14.4.5/544
438	Region Descriptor n, Word 2 (MPU_RGD3_WORD2)	32	R/W	See section	14.4.6/544
43C	Region Descriptor n, Word 3 (MPU_RGD3_WORD3)	32	R/W	0000_0000h	14.4.7/547
440	Region Descriptor n, Word 0 (MPU_RGD4_WORD0)	32	R/W	0000_0000h	14.4.4/543
444	Region Descriptor n, Word 1 (MPU_RGD4_WORD1)	32	R/W	See section	14.4.5/544
448	Region Descriptor n, Word 2 (MPU_RGD4_WORD2)	32	R/W	See section	14.4.6/544
44C	Region Descriptor n, Word 3 (MPU_RGD4_WORD3)	32	R/W	0000_0000h	14.4.7/547
450	Region Descriptor n, Word 0 (MPU_RGD5_WORD0)	32	R/W	0000_0000h	14.4.4/543
454	Region Descriptor n, Word 1 (MPU_RGD5_WORD1)	32	R/W	See section	14.4.5/544
458	Region Descriptor n, Word 2 (MPU_RGD5_WORD2)	32	R/W	See section	14.4.6/544
45C	Region Descriptor n, Word 3 (MPU_RGD5_WORD3)	32	R/W	0000_0000h	14.4.7/547
460	Region Descriptor n, Word 0 (MPU_RGD6_WORD0)	32	R/W	0000_0000h	14.4.4/543
464	Region Descriptor n, Word 1 (MPU_RGD6_WORD1)	32	R/W	See section	14.4.5/544
468	Region Descriptor n, Word 2 (MPU_RGD6_WORD2)	32	R/W	See section	14.4.6/544
46C	Region Descriptor n, Word 3 (MPU_RGD6_WORD3)	32	R/W	0000_0000h	14.4.7/547

Table continues on the next page...

MPU memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
470	Region Descriptor n, Word 0 (MPU_RGD7_WORD0)	32	R/W	0000_0000h	14.4.4/543
474	Region Descriptor n, Word 1 (MPU_RGD7_WORD1)	32	R/W	See section	14.4.5/544
478	Region Descriptor n, Word 2 (MPU_RGD7_WORD2)	32	R/W	See section	14.4.6/544
47C	Region Descriptor n, Word 3 (MPU_RGD7_WORD3)	32	R/W	0000_0000h	14.4.7/547
480	Region Descriptor n, Word 0 (MPU_RGD8_WORD0)	32	R/W	0000_0000h	14.4.4/543
484	Region Descriptor n, Word 1 (MPU_RGD8_WORD1)	32	R/W	See section	14.4.5/544
488	Region Descriptor n, Word 2 (MPU_RGD8_WORD2)	32	R/W	See section	14.4.6/544
48C	Region Descriptor n, Word 3 (MPU_RGD8_WORD3)	32	R/W	0000_0000h	14.4.7/547
490	Region Descriptor n, Word 0 (MPU_RGD9_WORD0)	32	R/W	0000_0000h	14.4.4/543
494	Region Descriptor n, Word 1 (MPU_RGD9_WORD1)	32	R/W	See section	14.4.5/544
498	Region Descriptor n, Word 2 (MPU_RGD9_WORD2)	32	R/W	See section	14.4.6/544
49C	Region Descriptor n, Word 3 (MPU_RGD9_WORD3)	32	R/W	0000_0000h	14.4.7/547
4A0	Region Descriptor n, Word 0 (MPU_RGD10_WORD0)	32	R/W	0000_0000h	14.4.4/543
4A4	Region Descriptor n, Word 1 (MPU_RGD10_WORD1)	32	R/W	See section	14.4.5/544
4A8	Region Descriptor n, Word 2 (MPU_RGD10_WORD2)	32	R/W	See section	14.4.6/544
4AC	Region Descriptor n, Word 3 (MPU_RGD10_WORD3)	32	R/W	0000_0000h	14.4.7/547
4B0	Region Descriptor n, Word 0 (MPU_RGD11_WORD0)	32	R/W	0000_0000h	14.4.4/543
4B4	Region Descriptor n, Word 1 (MPU_RGD11_WORD1)	32	R/W	See section	14.4.5/544
4B8	Region Descriptor n, Word 2 (MPU_RGD11_WORD2)	32	R/W	See section	14.4.6/544
4BC	Region Descriptor n, Word 3 (MPU_RGD11_WORD3)	32	R/W	0000_0000h	14.4.7/547
4C0	Region Descriptor n, Word 0 (MPU_RGD12_WORD0)	32	R/W	0000_0000h	14.4.4/543
4C4	Region Descriptor n, Word 1 (MPU_RGD12_WORD1)	32	R/W	See section	14.4.5/544
4C8	Region Descriptor n, Word 2 (MPU_RGD12_WORD2)	32	R/W	See section	14.4.6/544
4CC	Region Descriptor n, Word 3 (MPU_RGD12_WORD3)	32	R/W	0000_0000h	14.4.7/547
4D0	Region Descriptor n, Word 0 (MPU_RGD13_WORD0)	32	R/W	0000_0000h	14.4.4/543
4D4	Region Descriptor n, Word 1 (MPU_RGD13_WORD1)	32	R/W	See section	14.4.5/544
4D8	Region Descriptor n, Word 2 (MPU_RGD13_WORD2)	32	R/W	See section	14.4.6/544
4DC	Region Descriptor n, Word 3 (MPU_RGD13_WORD3)	32	R/W	0000_0000h	14.4.7/547
4E0	Region Descriptor n, Word 0 (MPU_RGD14_WORD0)	32	R/W	0000_0000h	14.4.4/543
4E4	Region Descriptor n, Word 1 (MPU_RGD14_WORD1)	32	R/W	See section	14.4.5/544
4E8	Region Descriptor n, Word 2 (MPU_RGD14_WORD2)	32	R/W	See section	14.4.6/544
4EC	Region Descriptor n, Word 3 (MPU_RGD14_WORD3)	32	R/W	0000_0000h	14.4.7/547
4F0	Region Descriptor n, Word 0 (MPU_RGD15_WORD0)	32	R/W	0000_0000h	14.4.4/543
4F4	Region Descriptor n, Word 1 (MPU_RGD15_WORD1)	32	R/W	See section	14.4.5/544
4F8	Region Descriptor n, Word 2 (MPU_RGD15_WORD2)	32	R/W	See section	14.4.6/544
4FC	Region Descriptor n, Word 3 (MPU_RGD15_WORD3)	32	R/W	0000_0000h	14.4.7/547
800	Region Descriptor Alternate Access Control n (MPU_RGDAAC0)	32	R/W	See section	14.4.8/548

Table continues on the next page...

MPU memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
804	Region Descriptor Alternate Access Control n (MPU_RGDAAC1)	32	R/W	See section	14.4.8/548
808	Region Descriptor Alternate Access Control n (MPU_RGDAAC2)	32	R/W	See section	14.4.8/548
80C	Region Descriptor Alternate Access Control n (MPU_RGDAAC3)	32	R/W	See section	14.4.8/548
810	Region Descriptor Alternate Access Control n (MPU_RGDAAC4)	32	R/W	See section	14.4.8/548
814	Region Descriptor Alternate Access Control n (MPU_RGDAAC5)	32	R/W	See section	14.4.8/548
818	Region Descriptor Alternate Access Control n (MPU_RGDAAC6)	32	R/W	See section	14.4.8/548
81C	Region Descriptor Alternate Access Control n (MPU_RGDAAC7)	32	R/W	See section	14.4.8/548
820	Region Descriptor Alternate Access Control n (MPU_RGDAAC8)	32	R/W	See section	14.4.8/548
824	Region Descriptor Alternate Access Control n (MPU_RGDAAC9)	32	R/W	See section	14.4.8/548
828	Region Descriptor Alternate Access Control n (MPU_RGDAAC10)	32	R/W	See section	14.4.8/548
82C	Region Descriptor Alternate Access Control n (MPU_RGDAAC11)	32	R/W	See section	14.4.8/548
830	Region Descriptor Alternate Access Control n (MPU_RGDAAC12)	32	R/W	See section	14.4.8/548
834	Region Descriptor Alternate Access Control n (MPU_RGDAAC13)	32	R/W	See section	14.4.8/548
838	Region Descriptor Alternate Access Control n (MPU_RGDAAC14)	32	R/W	See section	14.4.8/548
83C	Region Descriptor Alternate Access Control n (MPU_RGDAAC15)	32	R/W	See section	14.4.8/548

14.4.1 Control/Error Status Register (MPU_CESR)

Address: 0h base + 0h offset = 0h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	SPERR			0					1	0			HRL			
W	w1c															
Reset	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	1

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	
R	NSP				NRGD				0								VLD
W																	
Reset	*	*	*	*	0	0	1	0	0	0	0	0	0	0	0	0	

* Notes:

- NSP field: See the chip-specific MPU information for the reset value of this field.

MPU_CESR field descriptions

Field	Description
0–2 SPERR	Slave Port n Error Indicates a captured error in EARN and EDNRn. This bit is set when the hardware detects an error and records the faulting address and attributes. It is cleared by writing one to it. If another error is captured at the exact same cycle as the write, the flag remains set. A find-first-one instruction or equivalent can detect the presence of a captured error. The following shows the correspondence between the bit number and slave port number: • Bit 0 corresponds to slave port 0. • Bit 1 corresponds to slave port 1. • Bit 2 corresponds to slave port 2. 0 No error has occurred for slave port n. 1 An error has occurred for slave port n.
3–7 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
8 Reserved	This field is reserved. This read-only field is reserved and always has the value 1.
9–11 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
12–15 HRL	Hardware Revision Level Specifies the MPU's hardware and definition revision level. It can be read by software to determine the functional definition of the module.
16–19 NSP	Number Of Slave Ports Specifies the number of slave ports connected to the MPU.
20–23 NRGD	Number Of Region Descriptors Indicates the number of region descriptors implemented in the MPU.

Table continues on the next page...

MPU_CESR field descriptions (continued)

Field	Description
	0000 8 region descriptors 0001 12 region descriptors 0010 16 region descriptors
24–30 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
31 VLD	Valid Global enable/disable for the MPU. 0 MPU is disabled. All accesses from all bus masters are allowed. 1 MPU is enabled

14.4.2 Error Address Register, slave port n (MPU_EAR_n)

When the MPU detects an access error on slave port n, the 32-bit reference address is captured in this read-only register and the corresponding bit in CESR[SPERR_n] is set. Additional information about the faulting access is captured in the corresponding EDR_n at the same time. This register and the corresponding EDR_n contain the most recent access error; there are no hardware interlocks with CESR[SPERR_n], as the error registers are always loaded upon the occurrence of each protection violation.

Address: 0h base + 10h offset + (8d × i), where i=0d to 2d

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	EADDR																															
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	

MPU_EAR_n field descriptions

Field	Description
0–31 EADDR	Error Address Indicates the reference address from slave port n that generated the access error

14.4.3 Error Detail Register, slave port n (MPU_EDRn)

When the MPU detects an access error on slave port n, 32 bits of error detail are captured in this read-only register and the corresponding bit in CESR[SPERRn] is set. Information on the faulting address is captured in the corresponding EARN register at the same time. This register and the corresponding EARN register contain the most recent access error; there are no hardware interlocks with CESR[SPERRn] as the error registers are always loaded upon the occurrence of each protection violation.

Address: 0h base + 14h offset + (8d × i), where i=0d to 2d

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	EACD															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	EPID								EMN				EATTR			
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

MPU_EDRn field descriptions

Field	Description
0–15 EACD	Error Access Control Detail Indicates the region descriptor with the access error. • If EDRn contains a captured error and EACD is cleared, an access did not hit in any region descriptor. • If only a single EACD bit is set, the protection error was caused by a single non-overlapping region descriptor. • If two or more EACD bits are set, the protection error was caused by an overlapping set of region descriptors.
16–23 EPID	Error Process Identification Records the process identifier of the faulting reference. The process identifier is typically driven only by processor cores; for other bus masters, this field is cleared.
24–27 EMN	Error Master Number Indicates the bus master that generated the access error.
28–30 EATTR	Error Attributes Indicates attribute information about the faulting reference. NOTE: All other encodings are reserved. 000 User mode, instruction access 001 User mode, data access 010 Supervisor mode, instruction access 011 Supervisor mode, data access

Table continues on the next page...

MPU_EDR_n field descriptions (continued)

Field	Description
31 ERW	Error Read/Write Indicates the access type of the faulting reference. 0 Read 1 Write

14.4.4 Region Descriptor n, Word 0 (MPU_RGD_n_WORD0)

The first word of the region descriptor defines the 0-modulo-32 byte start address of the memory region. Writes to this register clear the region descriptor's valid bit (RGD_n_WORD3[VLD]).

Address: 0h base + 400h offset + (16d × i), where i=0d to 15d

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	SRTADDR																										0					
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	

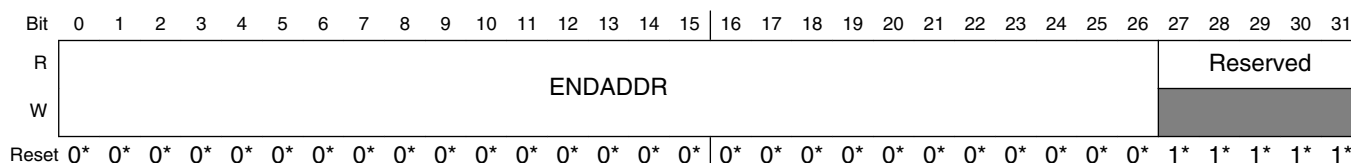
MPU_RGD_n_WORD0 field descriptions

Field	Description
0–26 SRTADDR	Start Address Defines the most significant bits of the 0-modulo-32 byte start address of the memory region.
27–31 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

14.4.5 Region Descriptor n, Word 1 (MPU_RGDn_WORD1)

The second word of the region descriptor defines the 31-modulo-32 byte end address of the memory region. Writes to this register clear the region descriptor's valid bit (RGDn_WORD3[VLD]).

Address: 0h base + 404h offset + (16d × i), where i=0d to 15d



* Notes:

- Reset value of RGD0_WORD1 is FFFF_FFFFh
- Reset value of RGD[1:15]_WORD1 is 0000_001Fh

MPU_RGDN_WORD1 field descriptions

Field	Description
0–26 ENDADDR	End Address Defines the most significant bits of the 31-modulo-32 byte end address of the memory region. NOTE: The MPU does not verify that ENDADDR ≥ SRTADDR.
27–31 Reserved	This field is reserved.

14.4.6 Region Descriptor n, Word 2 (MPU RGDn WORD2)

The third word of the region descriptor defines the access control rights of the memory region. The access control privileges depend on two broad classifications of bus masters:

- Bus masters 0–3 have a 5-bit field defining separate privilege rights for user and supervisor mode accesses. Each of these bus masters optionally includes a process identification field (if implemented for the master) within the definition.
- Bus masters 4–7 are limited to separate read and write permissions.

For the privilege rights of bus masters 0–3, there are three flags associated with this function:

- Read (r) refers to accessing the referenced memory address using an operand (data) fetch

- Write (w) refers to updating the referenced memory address using a store (data) instruction
- Execute (x) refers to reading the referenced memory address using an instruction fetch

Writes to RGDn_WORD2 clear the region descriptor's valid bit (RGDn_WORD3[VLD]). If only updating the access controls, write to RGDAACn instead because stores to these locations do not affect the descriptor's valid bit.

Address: 0h base + 408h offset + (16d × i), where i=0d to 15d

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	M7RE	M7WE	M6RE	M6WE	M5RE	M5WE	M4RE	M4WE	M3PE	M3SM		M3UM			M2PE	M2SM
W																
Reset	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	M2SM				M1PE		M1SM		M1UM		M0PE		M0SM		M0UM	
W																
Reset	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*

* Notes:

- Reset value of RGD0_WORD2 is 0061_8600h
Reset value of RGD[1:15]_WORD2 is 0000_0000h

MPU_RGDn_WORD2 field descriptions

Field	Description
0 M7RE	Bus Master 7 Read Enable 0 Bus master 7 reads terminate with an access error and the read is not performed 1 Bus master 7 reads allowed
1 M7WE	Bus Master 7 Write Enable 0 Bus master 7 writes terminate with an access error and the write is not performed 1 Bus master 7 writes allowed
2 M6RE	Bus Master 6 Read Enable 0 Bus master 6 reads terminate with an access error and the read is not performed 1 Bus master 6 reads allowed
3 M6WE	Bus Master 6 Write Enable 0 Bus master 6 writes terminate with an access error and the write is not performed 1 Bus master 6 writes allowed

Table continues on the next page...

MPU_RGDn_WORD2 field descriptions (continued)

Field	Description
4 M5RE	Bus Master 5 Read Enable 0 Bus master 5 reads terminate with an access error and the read is not performed 1 Bus master 5 reads allowed
5 M5WE	Bus Master 5 Write Enable 0 Bus master 5 writes terminate with an access error and the write is not performed 1 Bus master 5 writes allowed
6 M4RE	Bus Master 4 Read Enable 0 Bus master 4 reads terminate with an access error and the read is not performed 1 Bus master 4 reads allowed
7 M4WE	Bus Master 4 Write Enable 0 Bus master 4 writes terminate with an access error and the write is not performed 1 Bus master 4 writes allowed
8 M3PE	Bus Master 3 Process Identifier Enable 0 Do not include the process identifier in the evaluation 1 Include the process identifier and mask (RGDn_WORD3) in the region hit evaluation
9–10 M3SM	Bus Master 3 Supervisor Mode Access Control Defines the access controls for bus master 3 in Supervisor mode. 00 r/w/x; read, write and execute allowed 01 r/x; read and execute allowed, but no write 10 r/w; read and write allowed, but no execute 11 Same as User mode defined in M3UM
11–13 M3UM	Bus Master 3 User Mode Access Control Defines the access controls for bus master 3 in User mode. M3UM consists of three independent bits, enabling read (r), write (w), and execute (x) permissions. In M3UM[2:0]: M3UM[2] controls read permissions, M3UM[1] controls write permissions, and M3UM[0] controls execute permissions. 0 An attempted access of that mode may be terminated with an access error (if not allowed by another descriptor) and the access not performed. 1 Allows the given access type to occur
14 M2PE	Bus Master 2 Process Identifier Enable See M3PE description.
15–16 M2SM	Bus Master 2 Supervisor Mode Access Control See M3SM description.
17–19 M2UM	Bus Master 2 User Mode Access control See M3UM description.
20 M1PE	Bus Master 1 Process Identifier enable See M3PE description.

Table continues on the next page...

MPU_RGDn_WORD2 field descriptions (continued)

Field	Description
21–22 M1SM	Bus Master 1 Supervisor Mode Access Control See M3SM description.
23–25 M1UM	Bus Master 1 User Mode Access Control See M3UM description.
26 M0PE	Bus Master 0 Process Identifier enable See M0PE description.
27–28 M0SM	Bus Master 0 Supervisor Mode Access Control See M3SM description.
29–31 M0UM	Bus Master 0 User Mode Access Control See M3UM description.

14.4.7 Region Descriptor n, Word 3 (MPU_RGDn_WORD3)

The fourth word of the region descriptor contains the optional process identifier and mask, plus the region descriptor's valid bit.

Address: 0h base + 40Ch offset + (16d × i), where i=0d to 15d

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	PID							PIDMASK								
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0															VLD
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

MPU_RGDn_WORD3 field descriptions

Field	Description
0–7 PID	Process Identifier Specifies the process identifier that is included in the region hit determination if RGDn_WORD2[MxPE] is set. PIDMASK can mask individual bits in this field.
8–15 PIDMASK	Process Identifier Mask Provides a masking capability so that multiple process identifiers can be included as part of the region hit determination. If a bit in PIDMASK is set, then the corresponding PID bit is ignored in the comparison. This field and PID are included in the region hit determination if RGDn_WORD2[MxPE] is set. For more information on the handling of the PID and PIDMASK, see "Access Evaluation - Hit Determination."

Table continues on the next page...

MPU_RGDn_WORD3 field descriptions (continued)

Field	Description
16–30 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
31 VLD	Valid Signals the region descriptor is valid. Any write to RGDn_WORD0–2 clears this bit. 0 Region descriptor is invalid 1 Region descriptor is valid

14.4.8 Region Descriptor Alternate Access Control n (MPU_RGDAACn)

Because software may adjust only the access controls within a region descriptor (RGDn_WORD2) as different tasks execute, an alternate programming view of this 32-bit entity is available. Writing to this register does not affect the descriptor's valid bit.

Address: 0h base + 800h offset + (4d × i), where i=0d to 15d

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	M7RE	M7WE	M6RE	M6WE	M5RE	M5WE	M4RE	M4WE	M3PE	M3SM	M3UM	M2PE	M2SM			
W																
Reset	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*
Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	M2SM	M2UM	M1PE	M1SM	M1UM	M0PE	M0SM	M0UM								
W																
Reset	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*

* Notes:

- Reset value of RGDAAC0 is 0061_8600h
Reset value of RGDAAC[1:15] is 0000_0000h

MPU_RGDAACn field descriptions

Field	Description
0 M7RE	Bus Master 7 Read Enable

Table continues on the next page...

MPU_RGDAAC_n field descriptions (continued)

Field	Description
	0 Bus master 7 reads terminate with an access error and the read is not performed 1 Bus master 7 reads allowed
1 M7WE	Bus Master 7 Write Enable 0 Bus master 7 writes terminate with an access error and the write is not performed 1 Bus master 7 writes allowed
2 M6RE	Bus Master 6 Read Enable 0 Bus master 6 reads terminate with an access error and the read is not performed 1 Bus master 6 reads allowed
3 M6WE	Bus Master 6 Write Enable 0 Bus master 6 writes terminate with an access error and the write is not performed 1 Bus master 6 writes allowed
4 M5RE	Bus Master 5 Read Enable 0 Bus master 5 reads terminate with an access error and the read is not performed 1 Bus master 5 reads allowed
5 M5WE	Bus Master 5 Write Enable 0 Bus master 5 writes terminate with an access error and the write is not performed 1 Bus master 5 writes allowed
6 M4RE	Bus Master 4 Read Enable 0 Bus master 4 reads terminate with an access error and the read is not performed 1 Bus master 4 reads allowed
7 M4WE	Bus Master 4 Write Enable 0 Bus master 4 writes terminate with an access error and the write is not performed 1 Bus master 4 writes allowed
8 M3PE	Bus Master 3 Process Identifier Enable 0 Do not include the process identifier in the evaluation 1 Include the process identifier and mask (RGD _n .RGDAAC) in the region hit evaluation
9–10 M3SM	Bus Master 3 Supervisor Mode Access Control Defines the access controls for bus master 3 in Supervisor mode. 00 r/w/x; read, write and execute allowed 01 r/x; read and execute allowed, but no write 10 r/w; read and write allowed, but no execute 11 Same as User mode defined in M3UM
11–13 M3UM	Bus Master 3 User Mode Access Control Defines the access controls for bus master 3 in user mode. M3UM consists of three independent bits, enabling read (r), write (w), and execute (x) permissions. In M3UM[2:0]: M3UM[2] controls read permissions, M3UM[1] controls write permissions, and M3UM[0] controls execute permissions.

Table continues on the next page...

MPU_RGDAAC_n field descriptions (continued)

Field	Description
	0 An attempted access of that mode may be terminated with an access error (if not allowed by another descriptor) and the access not performed. 1 Allows the given access type to occur
14 M2PE	Bus Master 2 Process Identifier Enable See M3PE description.
15–16 M2SM	Bus Master 2 Supervisor Mode Access Control See M3SM description.
17–19 M2UM	Bus Master 2 User Mode Access Control See M3UM description.
20 M1PE	Bus Master 1 Process Identifier Enable See M3PE description.
21–22 M1SM	Bus Master 1 Supervisor Mode Access Control See M3SM description.
23–25 M1UM	Bus Master 1 User Mode Access Control See M3UM description.
26 M0PE	Bus Master 0 Process Identifier Enable See M3PE description.
27–28 M0SM	Bus Master 0 Supervisor Mode Access Control See M3SM description.
29–31 M0UM	Bus Master 0 User Mode Access Control See M3UM description.

14.5 Functional description

In this section, the functional operation of the MPU is detailed, including the operation of the access evaluation macro and the handling of error-terminated bus cycles.

14.5.1 Access evaluation macro

The basic operation of the MPU is performed in the access evaluation macro, a hardware structure replicated in the two-dimensional connection matrix. As shown in the following figure, the access evaluation macro inputs the crossbar bus address phase signals and the contents of a region descriptor (RGD_n) and performs two major functions:

- Region hit determination
- Detection of an access protection violation

The following figure shows a functional block diagram.

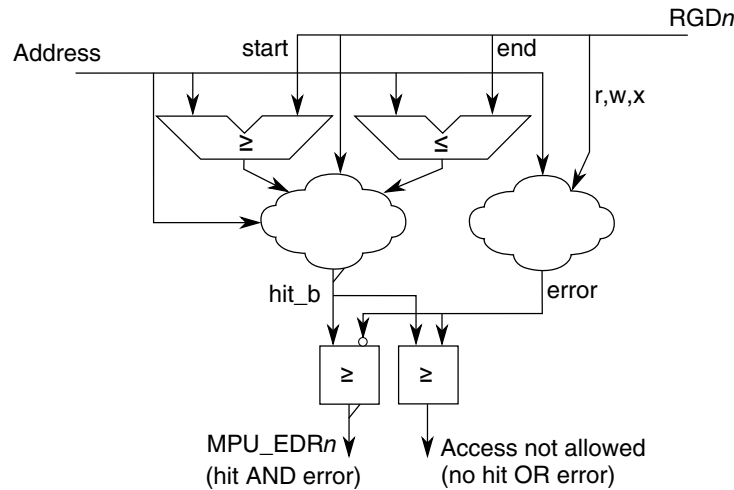


Figure 14-96. MPU access evaluation macro

14.5.1.1 Hit determination

To determine whether the current reference hits in the given region, two magnitude comparators are used with the region's start and end addresses. The boolean equation for this portion of the hit determination is:

```
region_hit = ((addr[31:5] >= RGDn_Word0[SRTADDR]) & (addr[31:5] <= RGDn_Word1[ENDADDR])) &
RGDn_Word3[VLD]
```

where *addr* is the current reference address, *RGDn_Word0[SRTADDR]* and *RGDn_Word1[ENDADDR]* are the start and end addresses, and *RGDn_Word3[VLD]* is the valid bit.

NOTE

The MPU does not verify that *ENDADDR* $\geq$ *SRTADDR*.

In addition to the comparison of the reference address versus the region descriptor's start and end addresses, the optional process identifier is examined against the region descriptor's PID and PIDMASK fields. A process identifier hit term is formed as follows:

```
pid_hit = ~RGDn_Word2[MxPE] |
((current_pid |
RGDn_Word3[PIDMASK]) == (RGDn_Word3[PID] | RGDn_Word3[PIDMASK]))
```

where the `current_pid` is the selected process identifier from the current bus master, and `RGD $n$ _Word3[PID]` and `RGD $n$ _Word3[PIDMASK]` are the process identifier fields from region descriptor n . For bus masters that do not output a process identifier, the MPU forces the `pid_hit` term to assert.

14.5.1.2 Privilege violation determination

While the access evaluation macro is determining region hit, the logic is also evaluating if the current access is allowed by the permissions defined in the region descriptor. Using the master and supervisor/user mode signals, a set of effective permissions is generated from the appropriate fields in the region descriptor. The protection violation logic then evaluates the access against the effective permissions using the specification shown below.

Table 14-100. Protection violation definition

Description	MxUM			Protection violation?
	r	w	x	
Instruction fetch read	—	—	0	Yes, no execute permission
	—	—	1	No, access is allowed
Data read	0	—	—	Yes, no read permission
	1	—	—	No, access is allowed
Data write	—	0	—	Yes, no write permission
	—	1	—	No, access is allowed

14.5.2 Putting it all together and error terminations

For each slave port monitored, the MPU performs a reduction-AND of all the individual terms from each access evaluation macro. This expression then terminates the bus cycle with an error and reports a protection error for three conditions:

- If the access does not hit in any region descriptor, a protection error is reported.
- If the access hits in a single region descriptor and that region signals a protection violation, a protection error is reported.
- If the access hits in multiple (overlapping) regions and all regions signal protection violations, a protection error is reported.

As shown in the third condition, granting permission is a higher priority than denying access for overlapping regions. This approach is more flexible to system software in region descriptor assignments. For an example of the use of overlapping region descriptors, see [Application information](#).

14.5.3 Power management

Disabling the MPU by clearing CESR[VLD] minimizes power dissipation. To minimize the power dissipation of an enabled MPU, invalidate unused region descriptors by clearing the associated RGDn_Word3[VLD] bits.

14.6 Initialization information

At system startup, load the appropriate number of region descriptors, including setting RGDn_Word3[VLD]. Setting CESR[VLD] enables the module.

If the system requires that all the loaded region descriptors be enabled simultaneously, first ensure that the entire MPU is disabled (CESR[VLD]=0).

Note

A region descriptor must be set to allow access to the MPU registers if further changes are needed.

14.7 Application information

In an operational system, interfacing with the MPU is generally classified into the following activities:

- Creating a new memory region—Load the appropriate region descriptor into an available RGDn, using four sequential 32-bit writes. The hardware assists in the maintenance of the valid bit, so if this approach is followed, there are no coherency issues with the multi-cycle descriptor writes. (Clearing RGDn_Word3[VLD] deletes/removes an existing memory region.)
- Altering only access privileges—To not affect the valid bit, write to the alternate version of the access control word (RGDAACn), so there are no coherency issues involved with the update. When the write completes, the memory region's access rights switch instantaneously to the new value.

- Changing a region's start and end addresses—Write a minimum of three words to the region descriptor (RGD n _Word{0,1,3}). Word 0 and 1 redefine the start and end addresses, respectively. Word 3 re-enables the region descriptor valid bit. In most situations, all four words of the region descriptor are rewritten.
- Accessing the MPU—Allocate a region descriptor to restrict MPU access to supervisor mode from a specific master.
- Detecting an access error—The current bus cycle is terminated with an error response and EAR n and EDR n capture information on the faulting reference. The error-terminated bus cycle typically initiates an error response in the originating bus master. For example, a processor core may respond with a bus error exception, while a data movement bus master may respond with an error interrupt. The processor can retrieve the captured error address and detail information simply by reading E{A,D}R n . CESR[SPERR] signals which error registers contain captured fault data.
- Overlapping region descriptors—Applying overlapping regions often reduces the number of descriptors required for a given set of access controls. In the overlapping memory space, the protection rights of the corresponding region descriptors are logically summed together (the boolean OR operator).

The following dual-core system example contains four bus masters:

- The two processors: CP0, CP1
- Two DMA engines: DMA1, a traditional data movement engine transferring data between RAM and peripherals and DMA2, a second engine transferring data to/from the RAM only

Consider the following region descriptor assignments:

Table 14-101. Overlapping region descriptor example

Region description	RGDn		CP0	CP1	DMA1	DMA2	
CP0 code	0		rwX	r--	—	—	Flash
CP1 code	1		r--	rwX	—	—	
CP0 data & stack	2		rw-	—	—	—	RAM
CP0 → CP1 shared data	2	3	r--	r--	—	—	
CP1 → CP0 shared data	4						
CP1 data & stack	4		—	rw-	—	—	
Shared DMA data	5		rw-	rw-	rw	rw	
MPU	6		rw-	rw-	—	—	Peripheral space
Peripherals	7		rw-	rw-	rw	—	

In this example, there are eight descriptors used to span nine regions in the three main spaces of the system memory map: flash, RAM, and peripheral space. Each region indicates the specific permissions for each of the four bus masters and this definition provides an appropriate set of shared, private and executable memory spaces.

Of particular interest are the two overlapping spaces: region descriptors 2 & 3 and 3 & 4.

The space defined by RGD2 with no overlap is a private data and stack area that provides read/write access to CP0 only. The overlapping space between RGD2 and RGD3 defines a shared data space for passing data from CP0 to CP1 and the access controls are defined by the logical OR of the two region descriptors. Thus, CP0 has $(rw- \mid r--) = (rw-)$ permissions, while CP1 has $(--- \mid r--) = (r--)$ permission in this space. Both DMA engines are excluded from this shared processor data region. The overlapping spaces between RGD3 and RGD4 defines another shared data space, this one for passing data from CP1 to CP0. For this overlapping space, CP0 has $(r-- \mid ---) = (r--)$ permission, while CP1 has $(rw- \mid r--) = (rw-)$ permission. The non-overlapped space of RGD4 defines a private data and stack area for CP1 only.

The space defined by RGD5 is a shared data region, accessible by all four bus masters. Finally, the slave peripheral space mapped onto the IPS bus is partitioned into two regions:

- One containing the MPU's programming model accessible only to the two processor cores
- The remaining peripheral region accessible to both processors and the traditional DMA1 master

This example shows one possible application of the capabilities of the MPU in a typical system.

Chapter 15

Interrupt Controller (INTC)

15.1 Chip-specific INTC information

The INTC on this chip supports two core processors. In the detailed INTC description, the cores are designated processor 0 and processor 1.

15.2 Introduction

The INTC:

- Provides priority-based preemptive scheduling of interrupt requests
- Schedules interrupt requests (IRQs) from software and internal peripherals to one or both processors (PRCs)
- Provides interrupt prioritization and preemption, interrupt masking, interrupt priority elevation, and protocol support

The INTC has two independent sets of priority arbitration/comparison, request selection, vector encoder and acknowledge logic—one set for each CPU. This arrangement allows each CPU to handle its software-assigned interrupt requests independently of the other CPU's operation, and it provides flexibility for the user to decide which core should handle which interrupt sources in the application. This flexibility comes from a set of configuration bits that allows any interrupt source to generate an interrupt request to Core 0, Core 1, or both cores.

For high-priority interrupt requests in these target applications, it is necessary to minimize the interval between the assertion of the interrupt request from the peripheral and the point at which the processor is performing useful work to service the interrupt request. The INTC supports this goal by providing a unique vector for each interrupt request source. It also provides 16 priorities so that lower priority ISRs do not delay the

execution of higher priority ISRs. Since each individual application will have different priorities for each source of interrupt request, the priority of each interrupt request is configurable.

When multiple tasks share a resource, coherent accesses to that resource need to be supported. The INTC supports the Priority Ceiling Protocol for coherent accesses. By providing a modifiable priority mask, the priority can be raised temporarily so that all tasks which share the resource are unable to preempt each other.

Multiple processors can assert interrupt requests to each other through software-settable interrupt requests. These same software-settable interrupt requests can also be used to separate the work involved in servicing an interrupt request into two parts, a high-priority portion and a low-priority portion. The high-priority portion is initiated by a peripheral interrupt request, but then the ISR can assert a software-settable interrupt request to finish the servicing in a lower priority ISR. Therefore these software-settable interrupt requests can be used instead of having the peripheral ISR schedule a task through the RTOS.

15.3 Block diagram

Interrupts implemented by the chip are defined in the *e200z759n3 Core Reference Manual*.

The block diagram for the INTC appears in the following figure.

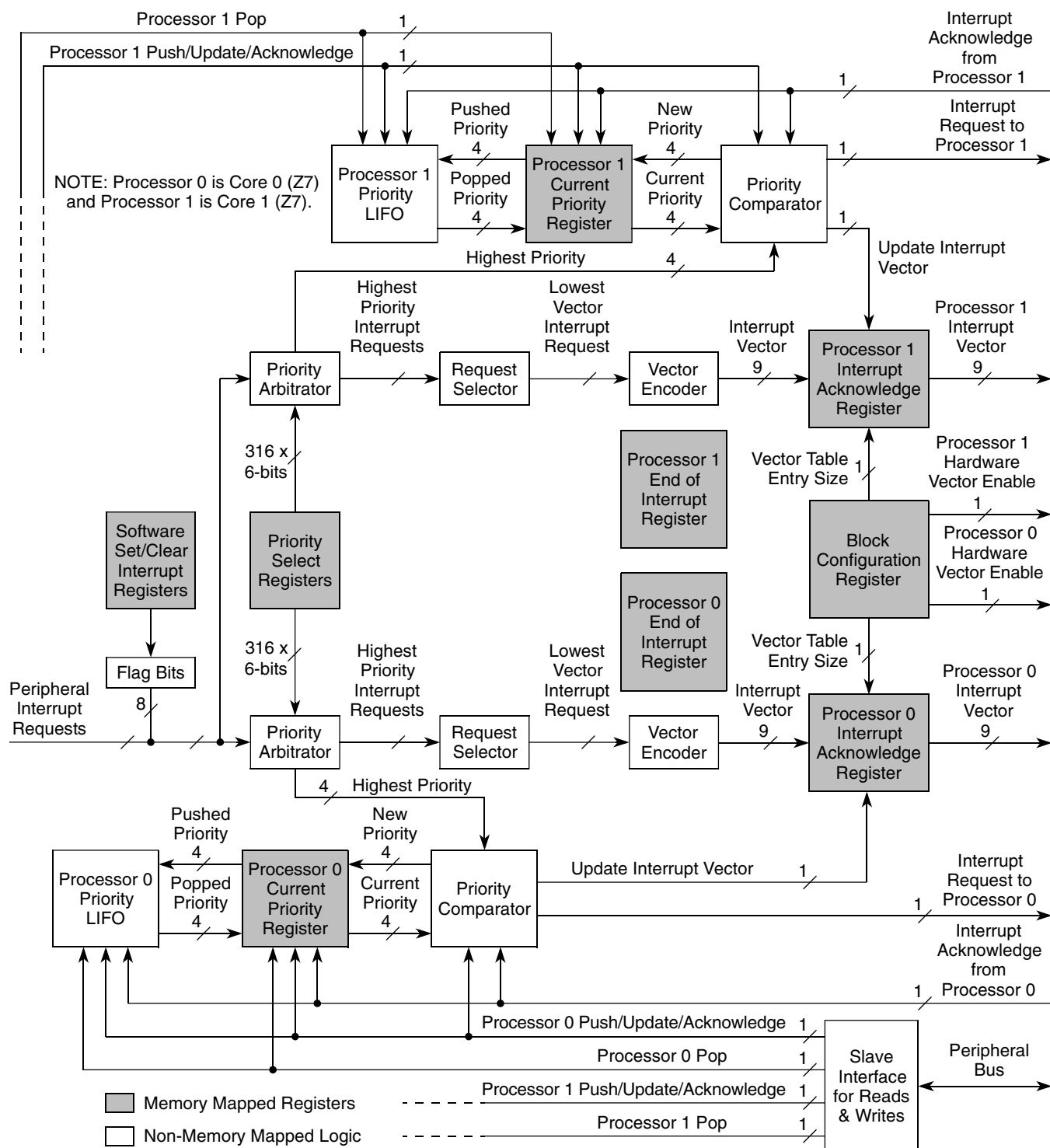


Figure 15-1. INTC block diagram

15.4 Features

- Each peripheral interrupt source is software-steerable to interrupt request outputs of processor 0, processor 1, or both
- 8 software-settable interrupt request sources
- 9-bit vector
 - Unique vector for each interrupt request source
 - Hardware connection to processor or read from register
- Each interrupt source can be programmed to one of 16 priorities
- Each interrupt source can be triggered by software
- Preemption
 - Preemptive prioritized interrupt requests to processor
 - ISR with higher priority preempts ISRs or tasks with lower priorities
 - Automatic pushing or popping of preempted priority to or from a LIFO
 - Ability to modify the ISR or task priority; modifying the priority can be used to implement the Priority Ceiling Protocol for accessing shared resources
- Low latency
 - Three INTC clock cycles from receipt of interrupt request from peripheral to interrupt request to processor

15.5 Modes of operation

The interrupt controller has two handshaking modes with the processor: software vector mode and hardware vector mode. The state of the hardware vector enable bit, `INTC_MCR[HVEN_PRC $n$ ]`, determines which mode is used for each CPU.

In debug mode the interrupt controller operation is identical to its normal operation of software vector mode or hardware vector mode.

15.5.1 Software vector mode

In software vector mode, as shown in the following figure, the CPU branches to a common interrupt exception handler whose location is determined by an address derived from special purpose registers IVPR and IVOR4. The interrupt exception handler reads the INTC_IACKR_PCR n to determine the vector of the interrupt request source.

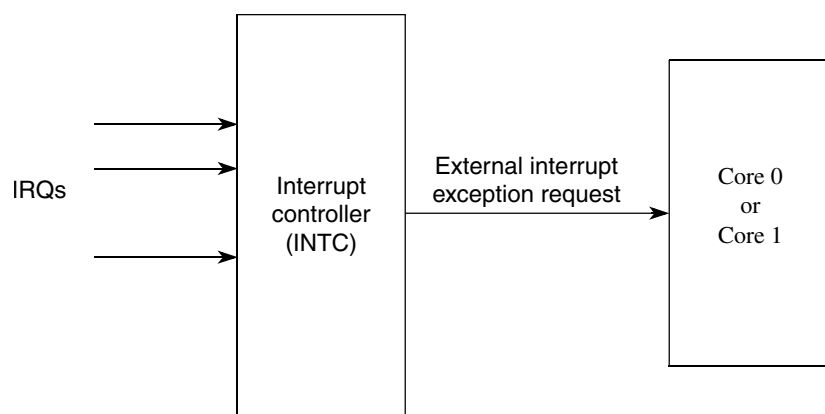
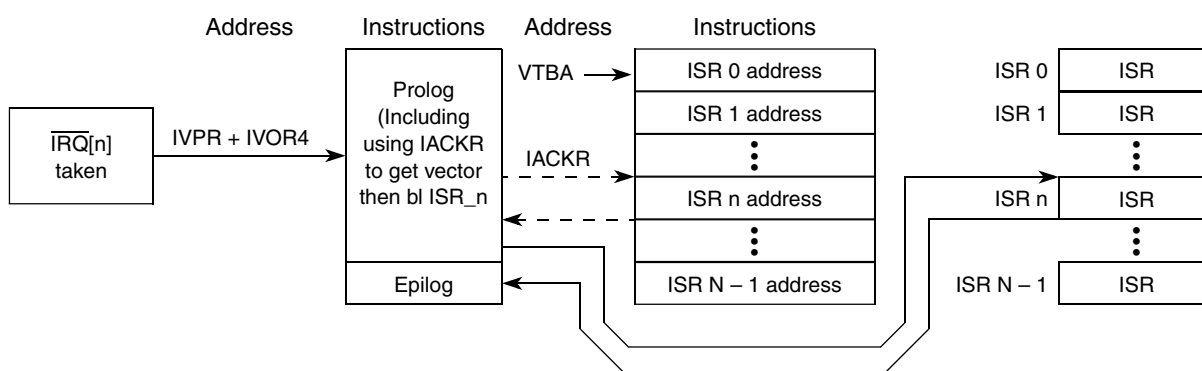


Figure 15-2. INTC software vector mode

The following figure shows the typical program flow for software vector mode.



N is the maximum number of usable interrupt vectors and includes eight software-settable IRQ vectors.

Figure 15-3. Program flow: software vector mode

The following figure shows how the common interrupt exception handler address is calculated by hardware. The upper half of the interrupt vector prefix register (IVPR) is added to the offset contained in the external input interrupt vector offset register (IVOR4). Because bits IVOR4[28:31] are not part of the offset value, the vector offset must be in a location in memory that is quad-word (16-byte) aligned.

Modes of operation

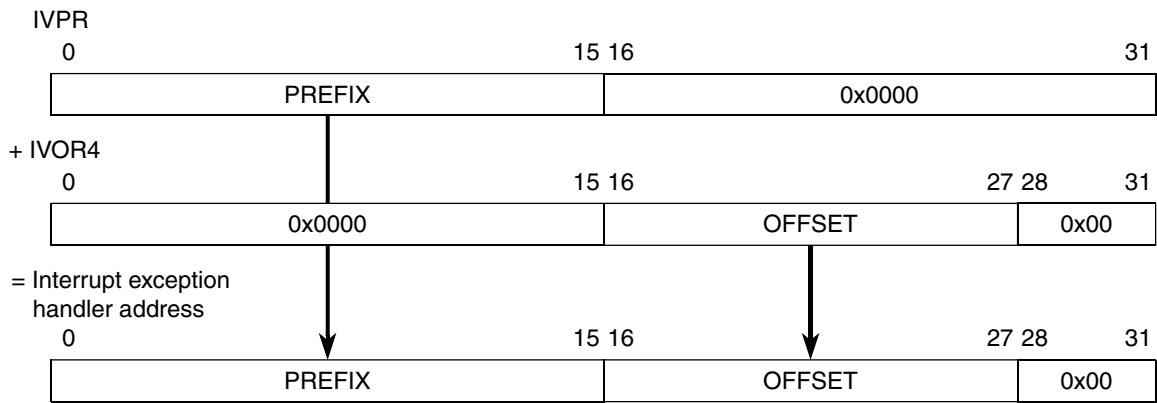


Figure 15-4. Software vector mode: interrupt exception handler address calculation

As shown in the preceding figure, the common interrupt exception handler reads the `INTC_IACKR_PRCn` to determine the vector of the interrupt request source. The `INTC_IACKR_PRCn` register contains a 32-bit address for a vector table base address (VTBA) plus an offset to access the interrupt vector (INTVEC). The address is then used to branch to the corresponding routine for that peripheral or software interrupt source.

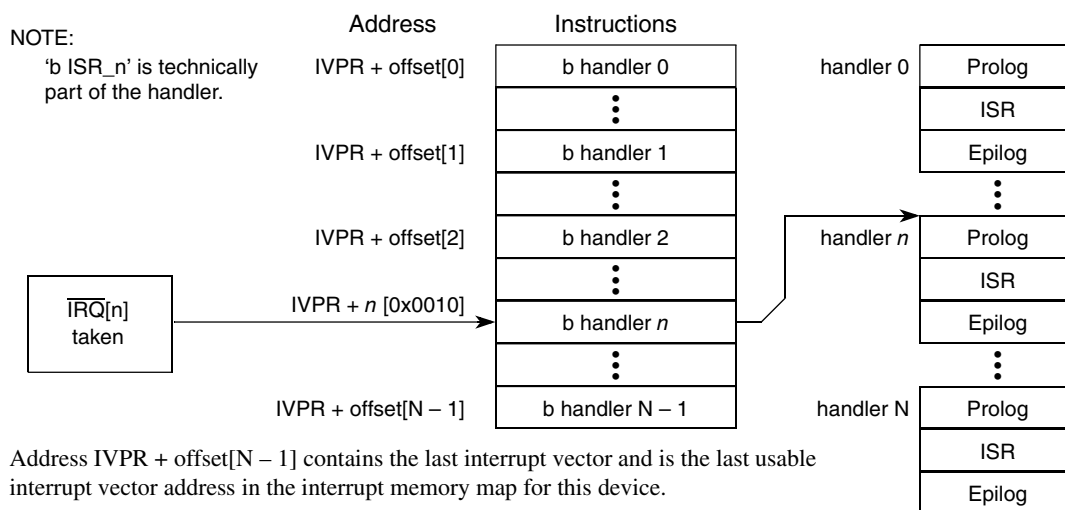
Reading the `INTC_IACKR_PRCn` acknowledges the INTC's interrupt request and negates the interrupt request to the processor. The interrupt request to the processor does not clear if a higher priority interrupt request arrives. Even in this case, INTVEC does not update to the higher priority request until the lower priority interrupt request is acknowledged by reading the `INTC_IACKR_PRCn`. The reading also pushes the PRI value in the INTC current priority register (`INTC_CPR_PRCn`) onto the LIFO and updates PRI in the `INTC_CPR_PRCn` with the priority of the interrupt request. The `INTC_CPR_PRCn` masks any peripheral or software settable interrupt request at the same or lower priority of the current value of the PRI field in `INTC_CPR_PRCn` from generating an interrupt request to the processor.

The interrupt exception handler must write to the end-of-interrupt register (`INTC_EOIR_PRCn`) to complete the operation. Writing to the `INTC_EOIR_PRCn` ends the servicing of the interrupt request. The INTC's LIFO is popped into the `INTC_CPR_PRCn`'s PRI field by writing to the `INTC_EOIR_PRCn`, and the size of a write does not affect the operation of the write. Those values and sizes written to this register neither update the `INTC_EOIR_PRCn` contents nor affect whether the LIFO pops. For possible future compatibility, write four bytes of all zeroes to the `INTC_EOIR_PRCn`. The timing relationship between popping the LIFO and disabling recognition of external input has no restriction. The writes can happen in either order.

However, disabling recognition of the external input before popping the LIFO eases the calculation of the maximum stack depth at the cost of postponing the servicing of the next interrupt request.

15.5.2 Hardware vector mode

The following figure shows the typical program flow for hardware vector mode.



N is the maximum number of usable interrupt vectors and includes eight software-settable IRQ vectors.

Figure 15-5. Program flow: hardware vector mode

In hardware vector mode, the interrupt exception handler address is specific to the peripheral or software-settable interrupt source rather than being common to all of them. No IVOR is used. The interrupt exception handler address is calculated by hardware as shown in the following figure. The upper half of the interrupt vector prefix register (IVPR) is added to an offset, which corresponds to the peripheral or software interrupt source which caused the interrupt request. The offset matches the value in the Interrupt Vector field, INTC_IACKR_PRCn[INTVEC]. Each interrupt exception handler address is aligned on a quad word (16-byte) boundary. IVOR4 is not used in this mode, and software does not need to read INTC_IACKR_PRCn to get the interrupt vector number.

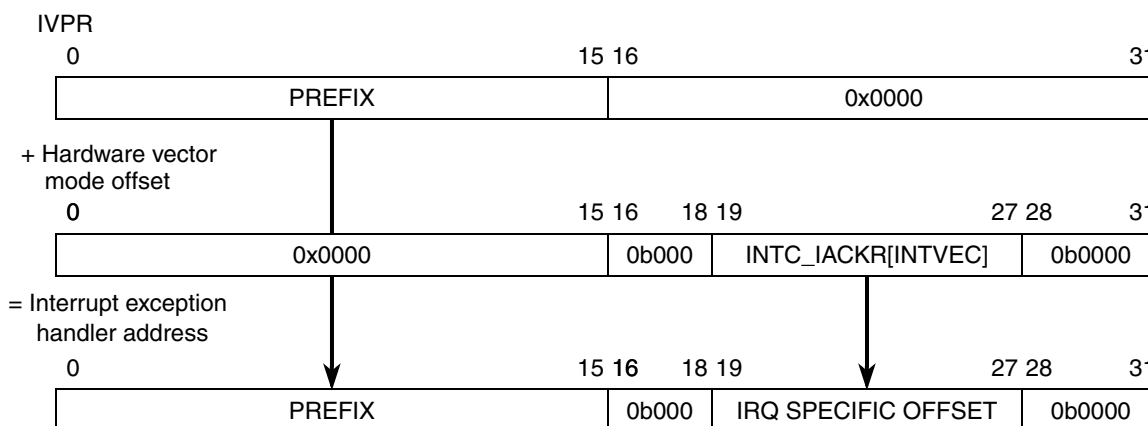


Figure 15-6. Hardware vector mode: interrupt exception handler address calculation

The processor negates INTC's interrupt request when automatically acknowledging the interrupt request. However, the interrupt request to the processor do not negate if a higher priority interrupt request arrives. Even in this case, the interrupt vector number does not update to the higher priority request until the lower priority request is acknowledged by the processor.

The assertion of the interrupt acknowledge signal pushes the PRI value in the INTC_CPR_PRC n onto the LIFO and updates PRI in the INTC_CPR_PRC n with the new priority.

15.6 External signal description

The INTC has no direct external MCU signals. However, there are external pins that can be configured in the SIU as external interrupt request input pins. When configured in this function, an interrupt on the pin sets an external interrupt flag. These flags can cause one of five peripheral interrupt requests to the interrupt controller.

15.7 Memory map and register definition

The actual availability and number of registers are chip-specific. For this information, see the chapter that describes how modules are configured and connected.

NOTE

For unimplemented locations in the memory map, read accesses return zeros and attempted write accesses are ignored.

The INTC register memory map is mirrored at 1 KB intervals. Referring to the following equation, the result of accessing any location when I is 1 to 15 is the same as the result of accessing a location when I is 0:

Offset + (400h x I), where I = 0 to 15

For example: PSR0 is accessed at 040h, 440h, 840h, and so on.

With the exception of the INTC_SSCIR n and INTC_PSR n registers, all of the registers are 32-bit. Any combination of accessing the four bytes of a register with a single access is supported, provided that the access does not cross a register boundary. These supported accesses include types and sizes of 8 bits, aligned 16 bits, misaligned 16 bits to the middle two bytes, and aligned 32 bits.

Although INTC_SSCIR n and INTC_PSR n are 8 bits wide, they can be accessed with a single 16-bit or 32-bit access, provided that the access does not cross a 32-bit boundary.

In software vector mode, the effects of a read of the INTC Interrupt Acknowledge register are the same regardless of the size of the read. In either software or hardware vector mode, the size of a write to the INTC End-of-Interrupt register does not affect the operation of the write.

INTC memory map

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
0	INTC Module Configuration Register (INTC_MCR)	32	R/W	0000_0000h	15.7.1/579
8	INTC Current Priority Register for Processor n (INTC_CPR_PRC0)	32	R/W	0000_000Fh	15.7.2/580
C	INTC Current Priority Register for Processor n (INTC_CPR_PRC1)	32	R/W	0000_000Fh	15.7.2/580
10	INTC Interrupt Acknowledge Register for Processor n (INTC_IACKR_PRC0)	32	R/W	0000_0000h	15.7.3/581
14	INTC Interrupt Acknowledge Register for Processor n (INTC_IACKR_PRC1)	32	R/W	0000_0000h	15.7.3/581
18	INTC End Of Interrupt Register for Processor n (INTC_EOIR_PRC0)	32	W	0000_0000h	15.7.4/582
1C	INTC End Of Interrupt Register for Processor n (INTC_EOIR_PRC1)	32	W	0000_0000h	15.7.4/582
20	INTC Software Set/Clear Interrupt Register (INTC_SSCIR0)	8	R/W	00h	15.7.5/583
21	INTC Software Set/Clear Interrupt Register (INTC_SSCIR1)	8	R/W	00h	15.7.5/583
22	INTC Software Set/Clear Interrupt Register (INTC_SSCIR2)	8	R/W	00h	15.7.5/583
23	INTC Software Set/Clear Interrupt Register (INTC_SSCIR3)	8	R/W	00h	15.7.5/583
24	INTC Software Set/Clear Interrupt Register (INTC_SSCIR4)	8	R/W	00h	15.7.5/583
25	INTC Software Set/Clear Interrupt Register (INTC_SSCIR5)	8	R/W	00h	15.7.5/583
26	INTC Software Set/Clear Interrupt Register (INTC_SSCIR6)	8	R/W	00h	15.7.5/583
27	INTC Software Set/Clear Interrupt Register (INTC_SSCIR7)	8	R/W	00h	15.7.5/583
40	INTC Priority Select Register (INTC_PSR0)	8	R/W	00h	15.7.6/584
41	INTC Priority Select Register (INTC_PSR1)	8	R/W	00h	15.7.6/584
42	INTC Priority Select Register (INTC_PSR2)	8	R/W	00h	15.7.6/584
43	INTC Priority Select Register (INTC_PSR3)	8	R/W	00h	15.7.6/584
44	INTC Priority Select Register (INTC_PSR4)	8	R/W	00h	15.7.6/584
45	INTC Priority Select Register (INTC_PSR5)	8	R/W	00h	15.7.6/584
46	INTC Priority Select Register (INTC_PSR6)	8	R/W	00h	15.7.6/584
47	INTC Priority Select Register (INTC_PSR7)	8	R/W	00h	15.7.6/584
48	INTC Priority Select Register (INTC_PSR8)	8	R/W	00h	15.7.6/584
49	INTC Priority Select Register (INTC_PSR9)	8	R/W	00h	15.7.6/584
4A	INTC Priority Select Register (INTC_PSR10)	8	R/W	00h	15.7.6/584
4B	INTC Priority Select Register (INTC_PSR11)	8	R/W	00h	15.7.6/584

Table continues on the next page...

INTC memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
4C	INTC Priority Select Register (INTC_PSR12)	8	R/W	00h	15.7.6/584
4D	INTC Priority Select Register (INTC_PSR13)	8	R/W	00h	15.7.6/584
4E	INTC Priority Select Register (INTC_PSR14)	8	R/W	00h	15.7.6/584
4F	INTC Priority Select Register (INTC_PSR15)	8	R/W	00h	15.7.6/584
50	INTC Priority Select Register (INTC_PSR16)	8	R/W	00h	15.7.6/584
51	INTC Priority Select Register (INTC_PSR17)	8	R/W	00h	15.7.6/584
52	INTC Priority Select Register (INTC_PSR18)	8	R/W	00h	15.7.6/584
53	INTC Priority Select Register (INTC_PSR19)	8	R/W	00h	15.7.6/584
54	INTC Priority Select Register (INTC_PSR20)	8	R/W	00h	15.7.6/584
55	INTC Priority Select Register (INTC_PSR21)	8	R/W	00h	15.7.6/584
56	INTC Priority Select Register (INTC_PSR22)	8	R/W	00h	15.7.6/584
57	INTC Priority Select Register (INTC_PSR23)	8	R/W	00h	15.7.6/584
58	INTC Priority Select Register (INTC_PSR24)	8	R/W	00h	15.7.6/584
59	INTC Priority Select Register (INTC_PSR25)	8	R/W	00h	15.7.6/584
5A	INTC Priority Select Register (INTC_PSR26)	8	R/W	00h	15.7.6/584
5B	INTC Priority Select Register (INTC_PSR27)	8	R/W	00h	15.7.6/584
5C	INTC Priority Select Register (INTC_PSR28)	8	R/W	00h	15.7.6/584
5D	INTC Priority Select Register (INTC_PSR29)	8	R/W	00h	15.7.6/584
5E	INTC Priority Select Register (INTC_PSR30)	8	R/W	00h	15.7.6/584
5F	INTC Priority Select Register (INTC_PSR31)	8	R/W	00h	15.7.6/584
60	INTC Priority Select Register (INTC_PSR32)	8	R/W	00h	15.7.6/584
61	INTC Priority Select Register (INTC_PSR33)	8	R/W	00h	15.7.6/584
62	INTC Priority Select Register (INTC_PSR34)	8	R/W	00h	15.7.6/584
63	INTC Priority Select Register (INTC_PSR35)	8	R/W	00h	15.7.6/584
64	INTC Priority Select Register (INTC_PSR36)	8	R/W	00h	15.7.6/584
65	INTC Priority Select Register (INTC_PSR37)	8	R/W	00h	15.7.6/584
66	INTC Priority Select Register (INTC_PSR38)	8	R/W	00h	15.7.6/584
67	INTC Priority Select Register (INTC_PSR39)	8	R/W	00h	15.7.6/584
68	INTC Priority Select Register (INTC_PSR40)	8	R/W	00h	15.7.6/584
69	INTC Priority Select Register (INTC_PSR41)	8	R/W	00h	15.7.6/584
6A	INTC Priority Select Register (INTC_PSR42)	8	R/W	00h	15.7.6/584
6B	INTC Priority Select Register (INTC_PSR43)	8	R/W	00h	15.7.6/584
6C	INTC Priority Select Register (INTC_PSR44)	8	R/W	00h	15.7.6/584
6D	INTC Priority Select Register (INTC_PSR45)	8	R/W	00h	15.7.6/584
6E	INTC Priority Select Register (INTC_PSR46)	8	R/W	00h	15.7.6/584
6F	INTC Priority Select Register (INTC_PSR47)	8	R/W	00h	15.7.6/584
70	INTC Priority Select Register (INTC_PSR48)	8	R/W	00h	15.7.6/584
71	INTC Priority Select Register (INTC_PSR49)	8	R/W	00h	15.7.6/584
72	INTC Priority Select Register (INTC_PSR50)	8	R/W	00h	15.7.6/584

Table continues on the next page...

INTC memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
73	INTC Priority Select Register (INTC_PSR51)	8	R/W	00h	15.7.6/584
74	INTC Priority Select Register (INTC_PSR52)	8	R/W	00h	15.7.6/584
75	INTC Priority Select Register (INTC_PSR53)	8	R/W	00h	15.7.6/584
76	INTC Priority Select Register (INTC_PSR54)	8	R/W	00h	15.7.6/584
77	INTC Priority Select Register (INTC_PSR55)	8	R/W	00h	15.7.6/584
78	INTC Priority Select Register (INTC_PSR56)	8	R/W	00h	15.7.6/584
79	INTC Priority Select Register (INTC_PSR57)	8	R/W	00h	15.7.6/584
7A	INTC Priority Select Register (INTC_PSR58)	8	R/W	00h	15.7.6/584
7B	INTC Priority Select Register (INTC_PSR59)	8	R/W	00h	15.7.6/584
7C	INTC Priority Select Register (INTC_PSR60)	8	R/W	00h	15.7.6/584
7D	INTC Priority Select Register (INTC_PSR61)	8	R/W	00h	15.7.6/584
7E	INTC Priority Select Register (INTC_PSR62)	8	R/W	00h	15.7.6/584
7F	INTC Priority Select Register (INTC_PSR63)	8	R/W	00h	15.7.6/584
80	INTC Priority Select Register (INTC_PSR64)	8	R/W	00h	15.7.6/584
81	INTC Priority Select Register (INTC_PSR65)	8	R/W	00h	15.7.6/584
82	INTC Priority Select Register (INTC_PSR66)	8	R/W	00h	15.7.6/584
83	INTC Priority Select Register (INTC_PSR67)	8	R/W	00h	15.7.6/584
84	INTC Priority Select Register (INTC_PSR68)	8	R/W	00h	15.7.6/584
85	INTC Priority Select Register (INTC_PSR69)	8	R/W	00h	15.7.6/584
86	INTC Priority Select Register (INTC_PSR70)	8	R/W	00h	15.7.6/584
87	INTC Priority Select Register (INTC_PSR71)	8	R/W	00h	15.7.6/584
88	INTC Priority Select Register (INTC_PSR72)	8	R/W	00h	15.7.6/584
89	INTC Priority Select Register (INTC_PSR73)	8	R/W	00h	15.7.6/584
8A	INTC Priority Select Register (INTC_PSR74)	8	R/W	00h	15.7.6/584
8B	INTC Priority Select Register (INTC_PSR75)	8	R/W	00h	15.7.6/584
8C	INTC Priority Select Register (INTC_PSR76)	8	R/W	00h	15.7.6/584
8D	INTC Priority Select Register (INTC_PSR77)	8	R/W	00h	15.7.6/584
8E	INTC Priority Select Register (INTC_PSR78)	8	R/W	00h	15.7.6/584
8F	INTC Priority Select Register (INTC_PSR79)	8	R/W	00h	15.7.6/584
90	INTC Priority Select Register (INTC_PSR80)	8	R/W	00h	15.7.6/584
91	INTC Priority Select Register (INTC_PSR81)	8	R/W	00h	15.7.6/584
92	INTC Priority Select Register (INTC_PSR82)	8	R/W	00h	15.7.6/584
93	INTC Priority Select Register (INTC_PSR83)	8	R/W	00h	15.7.6/584
94	INTC Priority Select Register (INTC_PSR84)	8	R/W	00h	15.7.6/584
95	INTC Priority Select Register (INTC_PSR85)	8	R/W	00h	15.7.6/584
96	INTC Priority Select Register (INTC_PSR86)	8	R/W	00h	15.7.6/584
97	INTC Priority Select Register (INTC_PSR87)	8	R/W	00h	15.7.6/584
98	INTC Priority Select Register (INTC_PSR88)	8	R/W	00h	15.7.6/584
99	INTC Priority Select Register (INTC_PSR89)	8	R/W	00h	15.7.6/584

Table continues on the next page...

INTC memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
9A	INTC Priority Select Register (INTC_PSR90)	8	R/W	00h	15.7.6/584
9B	INTC Priority Select Register (INTC_PSR91)	8	R/W	00h	15.7.6/584
9C	INTC Priority Select Register (INTC_PSR92)	8	R/W	00h	15.7.6/584
9D	INTC Priority Select Register (INTC_PSR93)	8	R/W	00h	15.7.6/584
9E	INTC Priority Select Register (INTC_PSR94)	8	R/W	00h	15.7.6/584
9F	INTC Priority Select Register (INTC_PSR95)	8	R/W	00h	15.7.6/584
A0	INTC Priority Select Register (INTC_PSR96)	8	R/W	00h	15.7.6/584
A1	INTC Priority Select Register (INTC_PSR97)	8	R/W	00h	15.7.6/584
A2	INTC Priority Select Register (INTC_PSR98)	8	R/W	00h	15.7.6/584
A3	INTC Priority Select Register (INTC_PSR99)	8	R/W	00h	15.7.6/584
A4	INTC Priority Select Register (INTC_PSR100)	8	R/W	00h	15.7.6/584
A5	INTC Priority Select Register (INTC_PSR101)	8	R/W	00h	15.7.6/584
A6	INTC Priority Select Register (INTC_PSR102)	8	R/W	00h	15.7.6/584
A7	INTC Priority Select Register (INTC_PSR103)	8	R/W	00h	15.7.6/584
A8	INTC Priority Select Register (INTC_PSR104)	8	R/W	00h	15.7.6/584
A9	INTC Priority Select Register (INTC_PSR105)	8	R/W	00h	15.7.6/584
AA	INTC Priority Select Register (INTC_PSR106)	8	R/W	00h	15.7.6/584
AB	INTC Priority Select Register (INTC_PSR107)	8	R/W	00h	15.7.6/584
AC	INTC Priority Select Register (INTC_PSR108)	8	R/W	00h	15.7.6/584
AD	INTC Priority Select Register (INTC_PSR109)	8	R/W	00h	15.7.6/584
AE	INTC Priority Select Register (INTC_PSR110)	8	R/W	00h	15.7.6/584
AF	INTC Priority Select Register (INTC_PSR111)	8	R/W	00h	15.7.6/584
B0	INTC Priority Select Register (INTC_PSR112)	8	R/W	00h	15.7.6/584
B1	INTC Priority Select Register (INTC_PSR113)	8	R/W	00h	15.7.6/584
B2	INTC Priority Select Register (INTC_PSR114)	8	R/W	00h	15.7.6/584
B3	INTC Priority Select Register (INTC_PSR115)	8	R/W	00h	15.7.6/584
B4	INTC Priority Select Register (INTC_PSR116)	8	R/W	00h	15.7.6/584
B5	INTC Priority Select Register (INTC_PSR117)	8	R/W	00h	15.7.6/584
B6	INTC Priority Select Register (INTC_PSR118)	8	R/W	00h	15.7.6/584
B7	INTC Priority Select Register (INTC_PSR119)	8	R/W	00h	15.7.6/584
B8	INTC Priority Select Register (INTC_PSR120)	8	R/W	00h	15.7.6/584
B9	INTC Priority Select Register (INTC_PSR121)	8	R/W	00h	15.7.6/584
BA	INTC Priority Select Register (INTC_PSR122)	8	R/W	00h	15.7.6/584
BB	INTC Priority Select Register (INTC_PSR123)	8	R/W	00h	15.7.6/584
BC	INTC Priority Select Register (INTC_PSR124)	8	R/W	00h	15.7.6/584
BD	INTC Priority Select Register (INTC_PSR125)	8	R/W	00h	15.7.6/584
BE	INTC Priority Select Register (INTC_PSR126)	8	R/W	00h	15.7.6/584
BF	INTC Priority Select Register (INTC_PSR127)	8	R/W	00h	15.7.6/584
C0	INTC Priority Select Register (INTC_PSR128)	8	R/W	00h	15.7.6/584

Table continues on the next page...

INTC memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
C1	INTC Priority Select Register (INTC_PSR129)	8	R/W	00h	15.7.6/584
C2	INTC Priority Select Register (INTC_PSR130)	8	R/W	00h	15.7.6/584
C3	INTC Priority Select Register (INTC_PSR131)	8	R/W	00h	15.7.6/584
C4	INTC Priority Select Register (INTC_PSR132)	8	R/W	00h	15.7.6/584
C5	INTC Priority Select Register (INTC_PSR133)	8	R/W	00h	15.7.6/584
C6	INTC Priority Select Register (INTC_PSR134)	8	R/W	00h	15.7.6/584
C7	INTC Priority Select Register (INTC_PSR135)	8	R/W	00h	15.7.6/584
C8	INTC Priority Select Register (INTC_PSR136)	8	R/W	00h	15.7.6/584
C9	INTC Priority Select Register (INTC_PSR137)	8	R/W	00h	15.7.6/584
CA	INTC Priority Select Register (INTC_PSR138)	8	R/W	00h	15.7.6/584
CB	INTC Priority Select Register (INTC_PSR139)	8	R/W	00h	15.7.6/584
CC	INTC Priority Select Register (INTC_PSR140)	8	R/W	00h	15.7.6/584
CD	INTC Priority Select Register (INTC_PSR141)	8	R/W	00h	15.7.6/584
CE	INTC Priority Select Register (INTC_PSR142)	8	R/W	00h	15.7.6/584
CF	INTC Priority Select Register (INTC_PSR143)	8	R/W	00h	15.7.6/584
D0	INTC Priority Select Register (INTC_PSR144)	8	R/W	00h	15.7.6/584
D1	INTC Priority Select Register (INTC_PSR145)	8	R/W	00h	15.7.6/584
D2	INTC Priority Select Register (INTC_PSR146)	8	R/W	00h	15.7.6/584
D3	INTC Priority Select Register (INTC_PSR147)	8	R/W	00h	15.7.6/584
D4	INTC Priority Select Register (INTC_PSR148)	8	R/W	00h	15.7.6/584
D5	INTC Priority Select Register (INTC_PSR149)	8	R/W	00h	15.7.6/584
D6	INTC Priority Select Register (INTC_PSR150)	8	R/W	00h	15.7.6/584
D7	INTC Priority Select Register (INTC_PSR151)	8	R/W	00h	15.7.6/584
D8	INTC Priority Select Register (INTC_PSR152)	8	R/W	00h	15.7.6/584
D9	INTC Priority Select Register (INTC_PSR153)	8	R/W	00h	15.7.6/584
DA	INTC Priority Select Register (INTC_PSR154)	8	R/W	00h	15.7.6/584
DB	INTC Priority Select Register (INTC_PSR155)	8	R/W	00h	15.7.6/584
DC	INTC Priority Select Register (INTC_PSR156)	8	R/W	00h	15.7.6/584
DD	INTC Priority Select Register (INTC_PSR157)	8	R/W	00h	15.7.6/584
DE	INTC Priority Select Register (INTC_PSR158)	8	R/W	00h	15.7.6/584
DF	INTC Priority Select Register (INTC_PSR159)	8	R/W	00h	15.7.6/584
E0	INTC Priority Select Register (INTC_PSR160)	8	R/W	00h	15.7.6/584
E1	INTC Priority Select Register (INTC_PSR161)	8	R/W	00h	15.7.6/584
E2	INTC Priority Select Register (INTC_PSR162)	8	R/W	00h	15.7.6/584
E3	INTC Priority Select Register (INTC_PSR163)	8	R/W	00h	15.7.6/584
E4	INTC Priority Select Register (INTC_PSR164)	8	R/W	00h	15.7.6/584
E5	INTC Priority Select Register (INTC_PSR165)	8	R/W	00h	15.7.6/584
E6	INTC Priority Select Register (INTC_PSR166)	8	R/W	00h	15.7.6/584
E7	INTC Priority Select Register (INTC_PSR167)	8	R/W	00h	15.7.6/584

Table continues on the next page...

INTC memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
E8	INTC Priority Select Register (INTC_PSR168)	8	R/W	00h	15.7.6/584
E9	INTC Priority Select Register (INTC_PSR169)	8	R/W	00h	15.7.6/584
EA	INTC Priority Select Register (INTC_PSR170)	8	R/W	00h	15.7.6/584
EB	INTC Priority Select Register (INTC_PSR171)	8	R/W	00h	15.7.6/584
EC	INTC Priority Select Register (INTC_PSR172)	8	R/W	00h	15.7.6/584
ED	INTC Priority Select Register (INTC_PSR173)	8	R/W	00h	15.7.6/584
EE	INTC Priority Select Register (INTC_PSR174)	8	R/W	00h	15.7.6/584
EF	INTC Priority Select Register (INTC_PSR175)	8	R/W	00h	15.7.6/584
F0	INTC Priority Select Register (INTC_PSR176)	8	R/W	00h	15.7.6/584
F1	INTC Priority Select Register (INTC_PSR177)	8	R/W	00h	15.7.6/584
F2	INTC Priority Select Register (INTC_PSR178)	8	R/W	00h	15.7.6/584
F3	INTC Priority Select Register (INTC_PSR179)	8	R/W	00h	15.7.6/584
F4	INTC Priority Select Register (INTC_PSR180)	8	R/W	00h	15.7.6/584
F5	INTC Priority Select Register (INTC_PSR181)	8	R/W	00h	15.7.6/584
F6	INTC Priority Select Register (INTC_PSR182)	8	R/W	00h	15.7.6/584
F7	INTC Priority Select Register (INTC_PSR183)	8	R/W	00h	15.7.6/584
F8	INTC Priority Select Register (INTC_PSR184)	8	R/W	00h	15.7.6/584
F9	INTC Priority Select Register (INTC_PSR185)	8	R/W	00h	15.7.6/584
FA	INTC Priority Select Register (INTC_PSR186)	8	R/W	00h	15.7.6/584
FB	INTC Priority Select Register (INTC_PSR187)	8	R/W	00h	15.7.6/584
FC	INTC Priority Select Register (INTC_PSR188)	8	R/W	00h	15.7.6/584
FD	INTC Priority Select Register (INTC_PSR189)	8	R/W	00h	15.7.6/584
FE	INTC Priority Select Register (INTC_PSR190)	8	R/W	00h	15.7.6/584
FF	INTC Priority Select Register (INTC_PSR191)	8	R/W	00h	15.7.6/584
100	INTC Priority Select Register (INTC_PSR192)	8	R/W	00h	15.7.6/584
101	INTC Priority Select Register (INTC_PSR193)	8	R/W	00h	15.7.6/584
102	INTC Priority Select Register (INTC_PSR194)	8	R/W	00h	15.7.6/584
103	INTC Priority Select Register (INTC_PSR195)	8	R/W	00h	15.7.6/584
104	INTC Priority Select Register (INTC_PSR196)	8	R/W	00h	15.7.6/584
105	INTC Priority Select Register (INTC_PSR197)	8	R/W	00h	15.7.6/584
106	INTC Priority Select Register (INTC_PSR198)	8	R/W	00h	15.7.6/584
107	INTC Priority Select Register (INTC_PSR199)	8	R/W	00h	15.7.6/584
108	INTC Priority Select Register (INTC_PSR200)	8	R/W	00h	15.7.6/584
109	INTC Priority Select Register (INTC_PSR201)	8	R/W	00h	15.7.6/584
10A	INTC Priority Select Register (INTC_PSR202)	8	R/W	00h	15.7.6/584
10B	INTC Priority Select Register (INTC_PSR203)	8	R/W	00h	15.7.6/584
10C	INTC Priority Select Register (INTC_PSR204)	8	R/W	00h	15.7.6/584
10D	INTC Priority Select Register (INTC_PSR205)	8	R/W	00h	15.7.6/584
10E	INTC Priority Select Register (INTC_PSR206)	8	R/W	00h	15.7.6/584

Table continues on the next page...

INTC memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
10F	INTC Priority Select Register (INTC_PSR207)	8	R/W	00h	15.7.6/584
110	INTC Priority Select Register (INTC_PSR208)	8	R/W	00h	15.7.6/584
111	INTC Priority Select Register (INTC_PSR209)	8	R/W	00h	15.7.6/584
112	INTC Priority Select Register (INTC_PSR210)	8	R/W	00h	15.7.6/584
113	INTC Priority Select Register (INTC_PSR211)	8	R/W	00h	15.7.6/584
114	INTC Priority Select Register (INTC_PSR212)	8	R/W	00h	15.7.6/584
115	INTC Priority Select Register (INTC_PSR213)	8	R/W	00h	15.7.6/584
116	INTC Priority Select Register (INTC_PSR214)	8	R/W	00h	15.7.6/584
117	INTC Priority Select Register (INTC_PSR215)	8	R/W	00h	15.7.6/584
118	INTC Priority Select Register (INTC_PSR216)	8	R/W	00h	15.7.6/584
119	INTC Priority Select Register (INTC_PSR217)	8	R/W	00h	15.7.6/584
11A	INTC Priority Select Register (INTC_PSR218)	8	R/W	00h	15.7.6/584
11B	INTC Priority Select Register (INTC_PSR219)	8	R/W	00h	15.7.6/584
11C	INTC Priority Select Register (INTC_PSR220)	8	R/W	00h	15.7.6/584
11D	INTC Priority Select Register (INTC_PSR221)	8	R/W	00h	15.7.6/584
11E	INTC Priority Select Register (INTC_PSR222)	8	R/W	00h	15.7.6/584
11F	INTC Priority Select Register (INTC_PSR223)	8	R/W	00h	15.7.6/584
120	INTC Priority Select Register (INTC_PSR224)	8	R/W	00h	15.7.6/584
121	INTC Priority Select Register (INTC_PSR225)	8	R/W	00h	15.7.6/584
122	INTC Priority Select Register (INTC_PSR226)	8	R/W	00h	15.7.6/584
123	INTC Priority Select Register (INTC_PSR227)	8	R/W	00h	15.7.6/584
124	INTC Priority Select Register (INTC_PSR228)	8	R/W	00h	15.7.6/584
125	INTC Priority Select Register (INTC_PSR229)	8	R/W	00h	15.7.6/584
126	INTC Priority Select Register (INTC_PSR230)	8	R/W	00h	15.7.6/584
127	INTC Priority Select Register (INTC_PSR231)	8	R/W	00h	15.7.6/584
128	INTC Priority Select Register (INTC_PSR232)	8	R/W	00h	15.7.6/584
129	INTC Priority Select Register (INTC_PSR233)	8	R/W	00h	15.7.6/584
12A	INTC Priority Select Register (INTC_PSR234)	8	R/W	00h	15.7.6/584
12B	INTC Priority Select Register (INTC_PSR235)	8	R/W	00h	15.7.6/584
12C	INTC Priority Select Register (INTC_PSR236)	8	R/W	00h	15.7.6/584
12D	INTC Priority Select Register (INTC_PSR237)	8	R/W	00h	15.7.6/584
12E	INTC Priority Select Register (INTC_PSR238)	8	R/W	00h	15.7.6/584
12F	INTC Priority Select Register (INTC_PSR239)	8	R/W	00h	15.7.6/584
130	INTC Priority Select Register (INTC_PSR240)	8	R/W	00h	15.7.6/584
131	INTC Priority Select Register (INTC_PSR241)	8	R/W	00h	15.7.6/584
132	INTC Priority Select Register (INTC_PSR242)	8	R/W	00h	15.7.6/584
133	INTC Priority Select Register (INTC_PSR243)	8	R/W	00h	15.7.6/584
134	INTC Priority Select Register (INTC_PSR244)	8	R/W	00h	15.7.6/584
135	INTC Priority Select Register (INTC_PSR245)	8	R/W	00h	15.7.6/584

Table continues on the next page...

INTC memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
136	INTC Priority Select Register (INTC_PSR246)	8	R/W	00h	15.7.6/584
137	INTC Priority Select Register (INTC_PSR247)	8	R/W	00h	15.7.6/584
138	INTC Priority Select Register (INTC_PSR248)	8	R/W	00h	15.7.6/584
139	INTC Priority Select Register (INTC_PSR249)	8	R/W	00h	15.7.6/584
13A	INTC Priority Select Register (INTC_PSR250)	8	R/W	00h	15.7.6/584
13B	INTC Priority Select Register (INTC_PSR251)	8	R/W	00h	15.7.6/584
13C	INTC Priority Select Register (INTC_PSR252)	8	R/W	00h	15.7.6/584
13D	INTC Priority Select Register (INTC_PSR253)	8	R/W	00h	15.7.6/584
13E	INTC Priority Select Register (INTC_PSR254)	8	R/W	00h	15.7.6/584
13F	INTC Priority Select Register (INTC_PSR255)	8	R/W	00h	15.7.6/584
140	INTC Priority Select Register (INTC_PSR256)	8	R/W	00h	15.7.6/584
141	INTC Priority Select Register (INTC_PSR257)	8	R/W	00h	15.7.6/584
142	INTC Priority Select Register (INTC_PSR258)	8	R/W	00h	15.7.6/584
143	INTC Priority Select Register (INTC_PSR259)	8	R/W	00h	15.7.6/584
144	INTC Priority Select Register (INTC_PSR260)	8	R/W	00h	15.7.6/584
145	INTC Priority Select Register (INTC_PSR261)	8	R/W	00h	15.7.6/584
146	INTC Priority Select Register (INTC_PSR262)	8	R/W	00h	15.7.6/584
147	INTC Priority Select Register (INTC_PSR263)	8	R/W	00h	15.7.6/584
148	INTC Priority Select Register (INTC_PSR264)	8	R/W	00h	15.7.6/584
149	INTC Priority Select Register (INTC_PSR265)	8	R/W	00h	15.7.6/584
14A	INTC Priority Select Register (INTC_PSR266)	8	R/W	00h	15.7.6/584
14B	INTC Priority Select Register (INTC_PSR267)	8	R/W	00h	15.7.6/584
14C	INTC Priority Select Register (INTC_PSR268)	8	R/W	00h	15.7.6/584
14D	INTC Priority Select Register (INTC_PSR269)	8	R/W	00h	15.7.6/584
14E	INTC Priority Select Register (INTC_PSR270)	8	R/W	00h	15.7.6/584
14F	INTC Priority Select Register (INTC_PSR271)	8	R/W	00h	15.7.6/584
150	INTC Priority Select Register (INTC_PSR272)	8	R/W	00h	15.7.6/584
151	INTC Priority Select Register (INTC_PSR273)	8	R/W	00h	15.7.6/584
152	INTC Priority Select Register (INTC_PSR274)	8	R/W	00h	15.7.6/584
153	INTC Priority Select Register (INTC_PSR275)	8	R/W	00h	15.7.6/584
154	INTC Priority Select Register (INTC_PSR276)	8	R/W	00h	15.7.6/584
155	INTC Priority Select Register (INTC_PSR277)	8	R/W	00h	15.7.6/584
156	INTC Priority Select Register (INTC_PSR278)	8	R/W	00h	15.7.6/584
157	INTC Priority Select Register (INTC_PSR279)	8	R/W	00h	15.7.6/584
158	INTC Priority Select Register (INTC_PSR280)	8	R/W	00h	15.7.6/584
159	INTC Priority Select Register (INTC_PSR281)	8	R/W	00h	15.7.6/584
15A	INTC Priority Select Register (INTC_PSR282)	8	R/W	00h	15.7.6/584
15B	INTC Priority Select Register (INTC_PSR283)	8	R/W	00h	15.7.6/584
15C	INTC Priority Select Register (INTC_PSR284)	8	R/W	00h	15.7.6/584

Table continues on the next page...

INTC memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
15D	INTC Priority Select Register (INTC_PSR285)	8	R/W	00h	15.7.6/584
15E	INTC Priority Select Register (INTC_PSR286)	8	R/W	00h	15.7.6/584
15F	INTC Priority Select Register (INTC_PSR287)	8	R/W	00h	15.7.6/584
160	INTC Priority Select Register (INTC_PSR288)	8	R/W	00h	15.7.6/584
161	INTC Priority Select Register (INTC_PSR289)	8	R/W	00h	15.7.6/584
162	INTC Priority Select Register (INTC_PSR290)	8	R/W	00h	15.7.6/584
163	INTC Priority Select Register (INTC_PSR291)	8	R/W	00h	15.7.6/584
164	INTC Priority Select Register (INTC_PSR292)	8	R/W	00h	15.7.6/584
165	INTC Priority Select Register (INTC_PSR293)	8	R/W	00h	15.7.6/584
166	INTC Priority Select Register (INTC_PSR294)	8	R/W	00h	15.7.6/584
167	INTC Priority Select Register (INTC_PSR295)	8	R/W	00h	15.7.6/584
168	INTC Priority Select Register (INTC_PSR296)	8	R/W	00h	15.7.6/584
169	INTC Priority Select Register (INTC_PSR297)	8	R/W	00h	15.7.6/584
16A	INTC Priority Select Register (INTC_PSR298)	8	R/W	00h	15.7.6/584
16B	INTC Priority Select Register (INTC_PSR299)	8	R/W	00h	15.7.6/584
16C	INTC Priority Select Register (INTC_PSR300)	8	R/W	00h	15.7.6/584
16D	INTC Priority Select Register (INTC_PSR301)	8	R/W	00h	15.7.6/584
16E	INTC Priority Select Register (INTC_PSR302)	8	R/W	00h	15.7.6/584
16F	INTC Priority Select Register (INTC_PSR303)	8	R/W	00h	15.7.6/584
170	INTC Priority Select Register (INTC_PSR304)	8	R/W	00h	15.7.6/584
171	INTC Priority Select Register (INTC_PSR305)	8	R/W	00h	15.7.6/584
172	INTC Priority Select Register (INTC_PSR306)	8	R/W	00h	15.7.6/584
173	INTC Priority Select Register (INTC_PSR307)	8	R/W	00h	15.7.6/584
174	INTC Priority Select Register (INTC_PSR308)	8	R/W	00h	15.7.6/584
175	INTC Priority Select Register (INTC_PSR309)	8	R/W	00h	15.7.6/584
176	INTC Priority Select Register (INTC_PSR310)	8	R/W	00h	15.7.6/584
177	INTC Priority Select Register (INTC_PSR311)	8	R/W	00h	15.7.6/584
178	INTC Priority Select Register (INTC_PSR312)	8	R/W	00h	15.7.6/584
179	INTC Priority Select Register (INTC_PSR313)	8	R/W	00h	15.7.6/584
17A	INTC Priority Select Register (INTC_PSR314)	8	R/W	00h	15.7.6/584
17B	INTC Priority Select Register (INTC_PSR315)	8	R/W	00h	15.7.6/584
17C	INTC Priority Select Register (INTC_PSR316)	8	R/W	00h	15.7.6/584
17D	INTC Priority Select Register (INTC_PSR317)	8	R/W	00h	15.7.6/584
17E	INTC Priority Select Register (INTC_PSR318)	8	R/W	00h	15.7.6/584
17F	INTC Priority Select Register (INTC_PSR319)	8	R/W	00h	15.7.6/584
180	INTC Priority Select Register (INTC_PSR320)	8	R/W	00h	15.7.6/584
181	INTC Priority Select Register (INTC_PSR321)	8	R/W	00h	15.7.6/584
182	INTC Priority Select Register (INTC_PSR322)	8	R/W	00h	15.7.6/584
183	INTC Priority Select Register (INTC_PSR323)	8	R/W	00h	15.7.6/584

Table continues on the next page...

INTC memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
184	INTC Priority Select Register (INTC_PSR324)	8	R/W	00h	15.7.6/584
185	INTC Priority Select Register (INTC_PSR325)	8	R/W	00h	15.7.6/584
186	INTC Priority Select Register (INTC_PSR326)	8	R/W	00h	15.7.6/584
187	INTC Priority Select Register (INTC_PSR327)	8	R/W	00h	15.7.6/584
188	INTC Priority Select Register (INTC_PSR328)	8	R/W	00h	15.7.6/584
189	INTC Priority Select Register (INTC_PSR329)	8	R/W	00h	15.7.6/584
18A	INTC Priority Select Register (INTC_PSR330)	8	R/W	00h	15.7.6/584
18B	INTC Priority Select Register (INTC_PSR331)	8	R/W	00h	15.7.6/584
18C	INTC Priority Select Register (INTC_PSR332)	8	R/W	00h	15.7.6/584
18D	INTC Priority Select Register (INTC_PSR333)	8	R/W	00h	15.7.6/584
18E	INTC Priority Select Register (INTC_PSR334)	8	R/W	00h	15.7.6/584
18F	INTC Priority Select Register (INTC_PSR335)	8	R/W	00h	15.7.6/584
190	INTC Priority Select Register (INTC_PSR336)	8	R/W	00h	15.7.6/584
191	INTC Priority Select Register (INTC_PSR337)	8	R/W	00h	15.7.6/584
192	INTC Priority Select Register (INTC_PSR338)	8	R/W	00h	15.7.6/584
193	INTC Priority Select Register (INTC_PSR339)	8	R/W	00h	15.7.6/584
194	INTC Priority Select Register (INTC_PSR340)	8	R/W	00h	15.7.6/584
195	INTC Priority Select Register (INTC_PSR341)	8	R/W	00h	15.7.6/584
196	INTC Priority Select Register (INTC_PSR342)	8	R/W	00h	15.7.6/584
197	INTC Priority Select Register (INTC_PSR343)	8	R/W	00h	15.7.6/584
198	INTC Priority Select Register (INTC_PSR344)	8	R/W	00h	15.7.6/584
199	INTC Priority Select Register (INTC_PSR345)	8	R/W	00h	15.7.6/584
19A	INTC Priority Select Register (INTC_PSR346)	8	R/W	00h	15.7.6/584
19B	INTC Priority Select Register (INTC_PSR347)	8	R/W	00h	15.7.6/584
19C	INTC Priority Select Register (INTC_PSR348)	8	R/W	00h	15.7.6/584
19D	INTC Priority Select Register (INTC_PSR349)	8	R/W	00h	15.7.6/584
19E	INTC Priority Select Register (INTC_PSR350)	8	R/W	00h	15.7.6/584
19F	INTC Priority Select Register (INTC_PSR351)	8	R/W	00h	15.7.6/584
1A0	INTC Priority Select Register (INTC_PSR352)	8	R/W	00h	15.7.6/584
1A1	INTC Priority Select Register (INTC_PSR353)	8	R/W	00h	15.7.6/584
1A2	INTC Priority Select Register (INTC_PSR354)	8	R/W	00h	15.7.6/584
1A3	INTC Priority Select Register (INTC_PSR355)	8	R/W	00h	15.7.6/584
1A4	INTC Priority Select Register (INTC_PSR356)	8	R/W	00h	15.7.6/584
1A5	INTC Priority Select Register (INTC_PSR357)	8	R/W	00h	15.7.6/584
1A6	INTC Priority Select Register (INTC_PSR358)	8	R/W	00h	15.7.6/584
1A7	INTC Priority Select Register (INTC_PSR359)	8	R/W	00h	15.7.6/584
1A8	INTC Priority Select Register (INTC_PSR360)	8	R/W	00h	15.7.6/584
1A9	INTC Priority Select Register (INTC_PSR361)	8	R/W	00h	15.7.6/584
1AA	INTC Priority Select Register (INTC_PSR362)	8	R/W	00h	15.7.6/584

Table continues on the next page...

INTC memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
1AB	INTC Priority Select Register (INTC_PSR363)	8	R/W	00h	15.7.6/584
1AC	INTC Priority Select Register (INTC_PSR364)	8	R/W	00h	15.7.6/584
1AD	INTC Priority Select Register (INTC_PSR365)	8	R/W	00h	15.7.6/584
1AE	INTC Priority Select Register (INTC_PSR366)	8	R/W	00h	15.7.6/584
1AF	INTC Priority Select Register (INTC_PSR367)	8	R/W	00h	15.7.6/584
1B0	INTC Priority Select Register (INTC_PSR368)	8	R/W	00h	15.7.6/584
1B1	INTC Priority Select Register (INTC_PSR369)	8	R/W	00h	15.7.6/584
1B2	INTC Priority Select Register (INTC_PSR370)	8	R/W	00h	15.7.6/584
1B3	INTC Priority Select Register (INTC_PSR371)	8	R/W	00h	15.7.6/584
1B4	INTC Priority Select Register (INTC_PSR372)	8	R/W	00h	15.7.6/584
1B5	INTC Priority Select Register (INTC_PSR373)	8	R/W	00h	15.7.6/584
1B6	INTC Priority Select Register (INTC_PSR374)	8	R/W	00h	15.7.6/584
1B7	INTC Priority Select Register (INTC_PSR375)	8	R/W	00h	15.7.6/584
1B8	INTC Priority Select Register (INTC_PSR376)	8	R/W	00h	15.7.6/584
1B9	INTC Priority Select Register (INTC_PSR377)	8	R/W	00h	15.7.6/584
1BA	INTC Priority Select Register (INTC_PSR378)	8	R/W	00h	15.7.6/584
1BB	INTC Priority Select Register (INTC_PSR379)	8	R/W	00h	15.7.6/584
1BC	INTC Priority Select Register (INTC_PSR380)	8	R/W	00h	15.7.6/584
1BD	INTC Priority Select Register (INTC_PSR381)	8	R/W	00h	15.7.6/584
1BE	INTC Priority Select Register (INTC_PSR382)	8	R/W	00h	15.7.6/584
1BF	INTC Priority Select Register (INTC_PSR383)	8	R/W	00h	15.7.6/584
1C0	INTC Priority Select Register (INTC_PSR384)	8	R/W	00h	15.7.6/584
1C1	INTC Priority Select Register (INTC_PSR385)	8	R/W	00h	15.7.6/584
1C2	INTC Priority Select Register (INTC_PSR386)	8	R/W	00h	15.7.6/584
1C3	INTC Priority Select Register (INTC_PSR387)	8	R/W	00h	15.7.6/584
1C4	INTC Priority Select Register (INTC_PSR388)	8	R/W	00h	15.7.6/584
1C5	INTC Priority Select Register (INTC_PSR389)	8	R/W	00h	15.7.6/584
1C6	INTC Priority Select Register (INTC_PSR390)	8	R/W	00h	15.7.6/584
1C7	INTC Priority Select Register (INTC_PSR391)	8	R/W	00h	15.7.6/584
1C8	INTC Priority Select Register (INTC_PSR392)	8	R/W	00h	15.7.6/584
1C9	INTC Priority Select Register (INTC_PSR393)	8	R/W	00h	15.7.6/584
1CA	INTC Priority Select Register (INTC_PSR394)	8	R/W	00h	15.7.6/584
1CB	INTC Priority Select Register (INTC_PSR395)	8	R/W	00h	15.7.6/584
1CC	INTC Priority Select Register (INTC_PSR396)	8	R/W	00h	15.7.6/584
1CD	INTC Priority Select Register (INTC_PSR397)	8	R/W	00h	15.7.6/584
1CE	INTC Priority Select Register (INTC_PSR398)	8	R/W	00h	15.7.6/584
1CF	INTC Priority Select Register (INTC_PSR399)	8	R/W	00h	15.7.6/584
1D0	INTC Priority Select Register (INTC_PSR400)	8	R/W	00h	15.7.6/584
1D1	INTC Priority Select Register (INTC_PSR401)	8	R/W	00h	15.7.6/584

Table continues on the next page...

INTC memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
1D2	INTC Priority Select Register (INTC_PSR402)	8	R/W	00h	15.7.6/584
1D3	INTC Priority Select Register (INTC_PSR403)	8	R/W	00h	15.7.6/584
1D4	INTC Priority Select Register (INTC_PSR404)	8	R/W	00h	15.7.6/584
1D5	INTC Priority Select Register (INTC_PSR405)	8	R/W	00h	15.7.6/584
1D6	INTC Priority Select Register (INTC_PSR406)	8	R/W	00h	15.7.6/584
1D7	INTC Priority Select Register (INTC_PSR407)	8	R/W	00h	15.7.6/584
1D8	INTC Priority Select Register (INTC_PSR408)	8	R/W	00h	15.7.6/584
1D9	INTC Priority Select Register (INTC_PSR409)	8	R/W	00h	15.7.6/584
1DA	INTC Priority Select Register (INTC_PSR410)	8	R/W	00h	15.7.6/584
1DB	INTC Priority Select Register (INTC_PSR411)	8	R/W	00h	15.7.6/584
1DC	INTC Priority Select Register (INTC_PSR412)	8	R/W	00h	15.7.6/584
1DD	INTC Priority Select Register (INTC_PSR413)	8	R/W	00h	15.7.6/584
1DE	INTC Priority Select Register (INTC_PSR414)	8	R/W	00h	15.7.6/584
1DF	INTC Priority Select Register (INTC_PSR415)	8	R/W	00h	15.7.6/584
1E0	INTC Priority Select Register (INTC_PSR416)	8	R/W	00h	15.7.6/584
1E1	INTC Priority Select Register (INTC_PSR417)	8	R/W	00h	15.7.6/584
1E2	INTC Priority Select Register (INTC_PSR418)	8	R/W	00h	15.7.6/584
1E3	INTC Priority Select Register (INTC_PSR419)	8	R/W	00h	15.7.6/584
1E4	INTC Priority Select Register (INTC_PSR420)	8	R/W	00h	15.7.6/584
1E5	INTC Priority Select Register (INTC_PSR421)	8	R/W	00h	15.7.6/584
1E6	INTC Priority Select Register (INTC_PSR422)	8	R/W	00h	15.7.6/584
1E7	INTC Priority Select Register (INTC_PSR423)	8	R/W	00h	15.7.6/584
1E8	INTC Priority Select Register (INTC_PSR424)	8	R/W	00h	15.7.6/584
1E9	INTC Priority Select Register (INTC_PSR425)	8	R/W	00h	15.7.6/584
1EA	INTC Priority Select Register (INTC_PSR426)	8	R/W	00h	15.7.6/584
1EB	INTC Priority Select Register (INTC_PSR427)	8	R/W	00h	15.7.6/584
1EC	INTC Priority Select Register (INTC_PSR428)	8	R/W	00h	15.7.6/584
1ED	INTC Priority Select Register (INTC_PSR429)	8	R/W	00h	15.7.6/584
1EE	INTC Priority Select Register (INTC_PSR430)	8	R/W	00h	15.7.6/584
1EF	INTC Priority Select Register (INTC_PSR431)	8	R/W	00h	15.7.6/584
1F0	INTC Priority Select Register (INTC_PSR432)	8	R/W	00h	15.7.6/584
1F1	INTC Priority Select Register (INTC_PSR433)	8	R/W	00h	15.7.6/584
1F2	INTC Priority Select Register (INTC_PSR434)	8	R/W	00h	15.7.6/584
1F3	INTC Priority Select Register (INTC_PSR435)	8	R/W	00h	15.7.6/584
1F4	INTC Priority Select Register (INTC_PSR436)	8	R/W	00h	15.7.6/584
1F5	INTC Priority Select Register (INTC_PSR437)	8	R/W	00h	15.7.6/584
1F6	INTC Priority Select Register (INTC_PSR438)	8	R/W	00h	15.7.6/584
1F7	INTC Priority Select Register (INTC_PSR439)	8	R/W	00h	15.7.6/584
1F8	INTC Priority Select Register (INTC_PSR440)	8	R/W	00h	15.7.6/584

Table continues on the next page...

INTC memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
1F9	INTC Priority Select Register (INTC_PSR441)	8	R/W	00h	15.7.6/584
1FA	INTC Priority Select Register (INTC_PSR442)	8	R/W	00h	15.7.6/584
1FB	INTC Priority Select Register (INTC_PSR443)	8	R/W	00h	15.7.6/584
1FC	INTC Priority Select Register (INTC_PSR444)	8	R/W	00h	15.7.6/584
1FD	INTC Priority Select Register (INTC_PSR445)	8	R/W	00h	15.7.6/584
1FE	INTC Priority Select Register (INTC_PSR446)	8	R/W	00h	15.7.6/584
1FF	INTC Priority Select Register (INTC_PSR447)	8	R/W	00h	15.7.6/584
200	INTC Priority Select Register (INTC_PSR448)	8	R/W	00h	15.7.6/584
201	INTC Priority Select Register (INTC_PSR449)	8	R/W	00h	15.7.6/584
202	INTC Priority Select Register (INTC_PSR450)	8	R/W	00h	15.7.6/584
203	INTC Priority Select Register (INTC_PSR451)	8	R/W	00h	15.7.6/584
204	INTC Priority Select Register (INTC_PSR452)	8	R/W	00h	15.7.6/584
205	INTC Priority Select Register (INTC_PSR453)	8	R/W	00h	15.7.6/584
206	INTC Priority Select Register (INTC_PSR454)	8	R/W	00h	15.7.6/584
207	INTC Priority Select Register (INTC_PSR455)	8	R/W	00h	15.7.6/584
208	INTC Priority Select Register (INTC_PSR456)	8	R/W	00h	15.7.6/584
209	INTC Priority Select Register (INTC_PSR457)	8	R/W	00h	15.7.6/584
20A	INTC Priority Select Register (INTC_PSR458)	8	R/W	00h	15.7.6/584
20B	INTC Priority Select Register (INTC_PSR459)	8	R/W	00h	15.7.6/584
20C	INTC Priority Select Register (INTC_PSR460)	8	R/W	00h	15.7.6/584
20D	INTC Priority Select Register (INTC_PSR461)	8	R/W	00h	15.7.6/584
20E	INTC Priority Select Register (INTC_PSR462)	8	R/W	00h	15.7.6/584
20F	INTC Priority Select Register (INTC_PSR463)	8	R/W	00h	15.7.6/584
210	INTC Priority Select Register (INTC_PSR464)	8	R/W	00h	15.7.6/584
211	INTC Priority Select Register (INTC_PSR465)	8	R/W	00h	15.7.6/584
212	INTC Priority Select Register (INTC_PSR466)	8	R/W	00h	15.7.6/584
213	INTC Priority Select Register (INTC_PSR467)	8	R/W	00h	15.7.6/584
214	INTC Priority Select Register (INTC_PSR468)	8	R/W	00h	15.7.6/584
215	INTC Priority Select Register (INTC_PSR469)	8	R/W	00h	15.7.6/584
216	INTC Priority Select Register (INTC_PSR470)	8	R/W	00h	15.7.6/584
217	INTC Priority Select Register (INTC_PSR471)	8	R/W	00h	15.7.6/584
218	INTC Priority Select Register (INTC_PSR472)	8	R/W	00h	15.7.6/584
219	INTC Priority Select Register (INTC_PSR473)	8	R/W	00h	15.7.6/584
21A	INTC Priority Select Register (INTC_PSR474)	8	R/W	00h	15.7.6/584
21B	INTC Priority Select Register (INTC_PSR475)	8	R/W	00h	15.7.6/584
21C	INTC Priority Select Register (INTC_PSR476)	8	R/W	00h	15.7.6/584
21D	INTC Priority Select Register (INTC_PSR477)	8	R/W	00h	15.7.6/584
21E	INTC Priority Select Register (INTC_PSR478)	8	R/W	00h	15.7.6/584
21F	INTC Priority Select Register (INTC_PSR479)	8	R/W	00h	15.7.6/584

Table continues on the next page...

INTC memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
220	INTC Priority Select Register (INTC_PSR480)	8	R/W	00h	15.7.6/584
221	INTC Priority Select Register (INTC_PSR481)	8	R/W	00h	15.7.6/584
222	INTC Priority Select Register (INTC_PSR482)	8	R/W	00h	15.7.6/584
223	INTC Priority Select Register (INTC_PSR483)	8	R/W	00h	15.7.6/584
224	INTC Priority Select Register (INTC_PSR484)	8	R/W	00h	15.7.6/584
225	INTC Priority Select Register (INTC_PSR485)	8	R/W	00h	15.7.6/584
226	INTC Priority Select Register (INTC_PSR486)	8	R/W	00h	15.7.6/584
227	INTC Priority Select Register (INTC_PSR487)	8	R/W	00h	15.7.6/584
228	INTC Priority Select Register (INTC_PSR488)	8	R/W	00h	15.7.6/584
229	INTC Priority Select Register (INTC_PSR489)	8	R/W	00h	15.7.6/584
22A	INTC Priority Select Register (INTC_PSR490)	8	R/W	00h	15.7.6/584
22B	INTC Priority Select Register (INTC_PSR491)	8	R/W	00h	15.7.6/584
22C	INTC Priority Select Register (INTC_PSR492)	8	R/W	00h	15.7.6/584
22D	INTC Priority Select Register (INTC_PSR493)	8	R/W	00h	15.7.6/584
22E	INTC Priority Select Register (INTC_PSR494)	8	R/W	00h	15.7.6/584
22F	INTC Priority Select Register (INTC_PSR495)	8	R/W	00h	15.7.6/584
230	INTC Priority Select Register (INTC_PSR496)	8	R/W	00h	15.7.6/584
231	INTC Priority Select Register (INTC_PSR497)	8	R/W	00h	15.7.6/584
232	INTC Priority Select Register (INTC_PSR498)	8	R/W	00h	15.7.6/584
233	INTC Priority Select Register (INTC_PSR499)	8	R/W	00h	15.7.6/584
234	INTC Priority Select Register (INTC_PSR500)	8	R/W	00h	15.7.6/584
235	INTC Priority Select Register (INTC_PSR501)	8	R/W	00h	15.7.6/584
236	INTC Priority Select Register (INTC_PSR502)	8	R/W	00h	15.7.6/584
237	INTC Priority Select Register (INTC_PSR503)	8	R/W	00h	15.7.6/584
238	INTC Priority Select Register (INTC_PSR504)	8	R/W	00h	15.7.6/584
239	INTC Priority Select Register (INTC_PSR505)	8	R/W	00h	15.7.6/584
23A	INTC Priority Select Register (INTC_PSR506)	8	R/W	00h	15.7.6/584
23B	INTC Priority Select Register (INTC_PSR507)	8	R/W	00h	15.7.6/584
23C	INTC Priority Select Register (INTC_PSR508)	8	R/W	00h	15.7.6/584
23D	INTC Priority Select Register (INTC_PSR509)	8	R/W	00h	15.7.6/584
23E	INTC Priority Select Register (INTC_PSR510)	8	R/W	00h	15.7.6/584
23F	INTC Priority Select Register (INTC_PSR511)	8	R/W	00h	15.7.6/584

15.7.1 INTC Module Configuration Register (INTC_MCR)

The Block Configuration Register is used to configure options of the INTC.

Address: 0h base + 0h offset = 0h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	VTES_PRC1		0				HVEN_PRC1	0		VTES_PRC0	0				HVEN_PRC0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

INTC_MCR field descriptions

Field	Description
0–17 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
18 VTES_PRC1	Vector Table Entry Size for Processor 1 (Core 1) For software vector mode only, controls the number of zeroes to the right of INTVEC_PRC1 in INTC_IACKR_PRC1. If the contents of INTC_IACKR_PRC1 are used as an address of an entry in a vector table, then the number of right-most zeroes determines the size of each vector table entry. 0 4 bytes 1 8 bytes
19–22 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
23 HVEN_PRC1	Hardware Vector Enable for Processor 1 (Core 1) Controls whether the INTC is in hardware vector mode or software vector mode. See Modes of operation for the details of the handshaking with the processor in each mode. 0 Software vector mode 1 Hardware vector mode
24–25 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
26 VTES_PRC0	Vector Table Entry Size for Processor 0 (Core 0)

Table continues on the next page...

INTC_MCR field descriptions (continued)

Field	Description
	For software vector mode only, controls the number of zeroes to the right of INTVEC_PRC0 in INTC_IACKR_PRC0. If the contents of INTC_IACKR_PRC0 are used as an address of an entry in a vector table, then the number of right-most zeroes determines the size of each vector table entry. 0 4 bytes 1 8 bytes
27–30 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
31 HVEN_PRC0	Hardware Vector Enable for Processor 0 (Core 0) Controls whether the INTC is in hardware vector mode or software vector mode. See Modes of operation for the details of the handshaking with the processor in each mode. 0 Software vector mode 1 Hardware vector mode

15.7.2 INTC Current Priority Register for Processor *n* (INTC_CPR_PRC_{*n*})

The INTC_CPR_PRC_{*n*} masks any peripheral or software-settable interrupt request that is set at the same or lower priority as the current value of the INTC_CPR_PRC_{*n*}[PRI] field from generating an interrupt request to the processor. When the INTC Interrupt Acknowledge register (INTC_IACKR_PRC_{*n*}) is read in software vector mode or the interrupt acknowledge signal from the processor is asserted in hardware vector mode, the value of PRI is pushed onto the LIFO, and PRI is updated with the priority of the preempting interrupt request. When the INTC End-of-Interrupt register (INTC_EOIR_PRC_{*n*}) is written, the LIFO is popped into the INTC_CPR_PRC_{*n*}[PRI] field. An exception case in hardware vector mode to this behavior is described in [Hardware vector mode](#).

The masking priority can be raised or lowered by writing to the PRI field, supporting the priority ceiling protocol. See [Priority ceiling protocol](#).

NOTE

The PRI register address space must be configured as guarded and cache-inhibited prior to writing to the register.

NOTE

A store to modify the PRI field that closely precedes or follows an access to a shared resource may result in a non-coherent access to that resource. See [Ensuring coherency](#) for example code to ensure coherency.

Address: 0h base + 8h offset + (4d × i), where i=0d to 1d

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0																											PRI				
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1

INTC_CPR_PRC_n field descriptions

Field	Description
0–27 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
28–31 PRI	Priority of the currently executing ISR, according to the field values 15 (highest priority) down to 0 (lowest priority).

15.7.3 INTC Interrupt Acknowledge Register for Processor *n* (INTC_IACKR_PRC_n)

The Interrupt Acknowledge Register for Processor *n* provides a value which can be used to load the address of an ISR from a vector table. The vector table can be composed of addresses of the ISRs specific to their respective interrupt vectors.

NOTE

Also, in software vector mode, the INTC_IACKR_PRC_n has side effects from reads. Therefore, it must not be read speculatively while in this mode. The side effects are the same regardless of the size of the read.

NOTE

Reading the INTC_IACKR_PRC_n has no side effects in hardware vector mode (when the corresponding INTC_MCR[HVEN_PRC_n] field is 1).

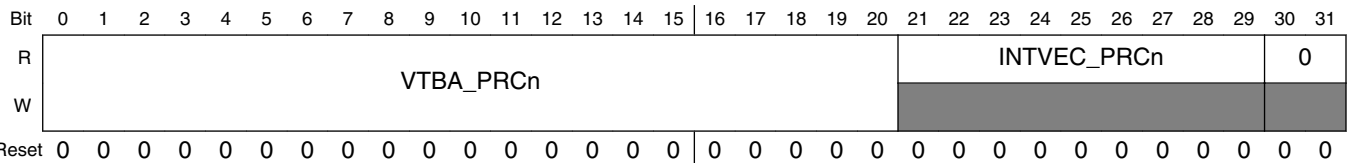
NOTE

In software vector mode, the INTC_IACKR_PRC_n must be read before setting MSR[EE]. No synchronization instruction is needed after reading the INTC_IACKR_PRC_n and before setting MSR[EE].

However, the time for the processor to recognize the assertion or negation of the external input to it is not defined by the book E architecture and can be greater than 0. Therefore, insert instructions between the reading of the INTC_IACKR_PRC_n and the setting of MSR[EE] that consumes at least two

processor clock cycles. This length of time allows the interrupt request negation to propagate through the processor before MSR[EE] is set.

Address: 0h base + 10h offset + (4d × i), where i=0d to 1d



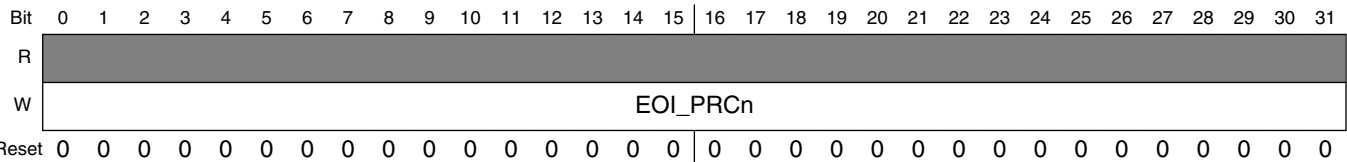
INTC_IACKR_PRCn field descriptions

Field	Description
0–20 VTBA_PRCn	Vector Table Base Address Can be the base address of a vector table of addresses of ISRs. VTBA_PRCn uses only the left-most 20 bits when the corresponding VTES_PRCn bit in INTC_MCR is asserted.
21–29 INTVEC_PRCn	Interrupt Vector Vector of the peripheral or software-settable interrupt request that caused the interrupt request to the processor. When the interrupt request to the processor asserts, the INTVEC is updated, whether the INTC is in software or hardware vector mode.
30–31 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

15.7.4 INTC End Of Interrupt Register for Processor n (INTC_EOIR_PRCn)

Writing to the INTC_EOIR_PRCn signals the end of the servicing of the interrupt request. When the INTC_EOIR_PRCn is written, the priority last pushed on the LIFO is popped into INTC_CPR_PRCn. The values and size of data written to the INTC_EOIR_PRCn are ignored. Those values and sizes written to this register neither update the INTC_EOIR_PRCn contents nor affect whether the LIFO pops. Typically, when you write to this register, write four all-zero bytes.

Address: 0h base + 18h offset + (4d × i), where i=0d to 1d



INTC_EOIR_PRCn field descriptions

Field	Description
0–31 EOI_PRCn	End of Interrupt Write four all-zero bytes to this field to signal the end of the servicing of an interrupt request.

15.7.5 INTC Software Set/Clear Interrupt Register (INTC_SSCIRn)

The INTC_SSCIR registers support the setting or clearing of software-settable interrupt requests. These registers contain independent sets of bits to set and clear a corresponding flag bit by software. With the exception of being set by software, this flag bit behaves the same as a flag bit set within a peripheral. This flag bit generates an interrupt request within the INTC just like a peripheral interrupt request. Writing a 1 to SETn will leave SETn unchanged at 0 but will set CLRn. Writing a 0 to SETn will have no effect. CLRn is the flag bit. Writing a 1 to CLRn will clear it. Writing a 0 to CLRn will have no effect. If a 1 is written to a pair of SETn and CLRn bits at the same time, CLRn is asserted, regardless of whether CLRn was asserted before the write.

Address: 0h base + 20h offset + (1d × i), where i=0d to 7d

Bit	0	1	2	3	4	5	6	7
Read	0						0	CLRn
Write							SETn	w1c
Reset	0	0	0	0	0	0	0	0

INTC_SSCIRn field descriptions

Field	Description
0–5 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
6 SETn	Set flag bits. Writing a 1 will set the corresponding CLRn bit. Writing a 0 will have no effect. Each SETn is read as a 0.
7 CLRn	Clear flag bits. CLRn is the flag bit. Writing a 1 to CLRn will clear it provided that a 1 is not written simultaneously to its corresponding SETn bit. Writing a 0 to CLRn will have no effect. 0 Interrupt request not pending within INTC. 1 Interrupt request pending within INTC.

15.7.6 INTC Priority Select Register (INTC_PSRn)

The Priority Select Registers support the selection of an individual priority for each source of interrupt request, and the routing of the request to one or both processors. The unique vector of each peripheral or software-settable interrupt request determines which INTC_PSRn is assigned to that interrupt request. The software-settable interrupt requests 0–7 are assigned vectors 0–7, and their priorities are configured in INTC_PSR0 to INTC_PSR7, respectively. The peripheral interrupt requests are assigned to vectors in the range 8 to 511, and their priorities are configured in the range of registers INTC_PSR8 to INTC_PSR511, respectively.

NOTE

Interrupts are not detected while the PSRs are being written.
The PRC_SELn or PRIn field of an INTC_PSRn must not be modified while the corresponding peripheral or software settable interrupt request is asserted.

Address: 0h base + 40h offset + (1d × i), where i=0d to 511d



INTC_PSRn field descriptions

Field	Description
0–1 PRC_SELn	Processor select bits. If an interrupt source is enabled, PRC_SELn[3:0] selects whether the interrupt request is to be sent to processor 0, processor 1, or both. 00 Interrupt request sent to processor 0 01 Interrupt request sent to both processors 10 Reserved 11 Interrupt request sent to processor 1
2–3 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
4–7 PRIn	Priority select. Selects the priority for the interrupt requests. PRI has values 15 (highest priority) down to 0 (lowest priority).

15.8 Functional description

The functional description involves the areas of interrupt request sources, priority management, and handshaking with the processor. In addition, spaces in the memory map have been reserved for other possible implementations of the INTC.

15.8.1 Interrupt request sources

The INTC has two types of interrupt requests, peripheral and software-settable. These interrupt requests can assert on any clock cycle.

The INTC has no spurious vector support. Therefore, if an asserted peripheral or software-settable interrupt request, whose PRI_n value in $INTC\_PSR_n$ is higher than the PRI value in $INTC\_CPR_n$, negates before the interrupt request to the processor for that peripheral or software-settable interrupt request is acknowledged, then the interrupt request to the processor still can assert (or will remain asserted) for that peripheral or software-settable interrupt request. In this case, the interrupt vector will correspond to that peripheral or software-settable interrupt request. Also, the PRI value in the associated $INTC\_CPR_n$ is updated with the corresponding PRI_n value in $INTC\_PSR_n$.

Furthermore, clearing the peripheral interrupt request's enable bit in the peripheral, or (alternatively) setting its mask bit, has the same consequences as clearing its flag bit. Setting its enable bit or clearing its mask bit while its flag bit is asserted has the same effect on the INTC as an interrupt event setting the flag bit.

15.8.1.1 Peripheral interrupt requests

An interrupt event in a peripheral's hardware sets a flag bit which resides in that peripheral. The interrupt request from the peripheral is driven by that flag bit.

The time from when the peripheral starts to drive its peripheral interrupt request to the INTC, to the time that the INTC starts to drive the interrupt request to the processor, is three clocks.

Interrupt requests from devices external to this chip are classified as peripheral interrupt requests in this document. This type of external peripheral interrupts is handled by the SIU.

15.8.1.2 Software-settable interrupt requests

The software set/clear interrupt registers ($INTC\_SSCIR_n$) support the setting or clearing of software-settable interrupt requests. These registers contain independent sets of bits to set and clear a corresponding flag bit by software. An interrupt request is triggered by software by writing a 1 to a SET_n bit in $INTC\_SSCIR_n$. This write sets the corresponding CLR_n bit, which is a flag bit, resulting in the interrupt request. The interrupt request is cleared by writing a 1 to the CLR_n bit. Specific behavior includes the following:

- Writing a 1 to SET n leaves SET n unchanged at 0 but sets the flag bit (which is the CLR n bit).
- Writing a 0 to SET n has no effect.
- Writing a 1 to CLR n clears the flag (CLR n) bit.
- Writing a 0 to CLR n has no effect.
- If a 1 is written to a pair of SET n and CLR n bits at the same time, the flag (CLR n) is set, regardless of whether CLR n was asserted before the write.

The time from the write to the SET n bit, to the time that the INTC starts to drive the interrupt request to the processor, is four clocks.

15.8.1.3 Unique vector for each interrupt request source

Each peripheral and software-settable interrupt request is assigned a hardwired unique 9-bit vector. Software-settable interrupts 0–7 are assigned vectors 0–7, respectively. The peripheral interrupt requests are assigned vectors from 8 to a number as high as needed to cover all peripheral interrupt requests.

15.8.2 Priority management

The asserted interrupt requests are compared to each other based on their PRI n and PRC_SEL n values set in INTC_PSR n . The result of that comparison also is compared to PRI in the associated INTC_CPR n . The results of those comparisons are used to manage the priority of the ISR being executed by the associated processor. The associated LIFO also assists in managing that priority.

15.8.2.1 Current priority and preemption

The priority arbitrator, selector, encoder, and comparator logic shown in [Figure 15-1](#) are used to compare the priority of the asserted interrupt requests to the current priority. If the priority of any asserted peripheral or software-settable interrupt request is higher than the current priority for a given processor, then the interrupt request to the processor is asserted. Also, a unique vector for the preempting peripheral or software-settable interrupt request is generated for the associated INTC_IACKR n , and if in hardware vector mode, for the interrupt vector provided to the processor.

15.8.2.1.1 Priority arbitrator

The priority arbitrator for each processor compares all the priorities of all of the asserted interrupt requests assigned to that processor, both peripheral and software-settable. The output of the priority arbitrator is the highest of those priorities assigned to a given processor. Also, any interrupt requests which have this highest priority are output as asserted interrupt requests to the associated selector logic.

15.8.2.1.2 Selector

If only one interrupt request from the associated priority arbitrator is asserted, then it is passed as asserted to the associated encoder logic. If multiple interrupt requests from the associated priority arbitrator are asserted, then only the one with the lowest vector is passed as asserted to the associated encoder logic. The lower vector is chosen regardless of the time order of the assertions of the peripheral or software-settable interrupt requests.

15.8.2.1.3 Encoder

The encoder logic generates the unique 9-bit vector for the asserted interrupt request from the request selector submodule for the associated processor.

15.8.2.1.4 Comparator

The comparator logic compares the highest priority output from the associated priority arbitrator subblock with PRI in the associated INTC_CPR_n. If the comparator detects that this highest priority is higher than the current priority, then it asserts the interrupt request to the associated processor. This interrupt request to the processor asserts whether this highest priority is raised above the value of PRI in the associated INTC_CPR_n, or the PRI value in the associated INTC_CPR_n is lowered below this highest priority. This highest priority then becomes the new priority which is written to PRI in the associated INTC_CPR_n when the interrupt request to the processor is acknowledged. Interrupt requests whose PRI_n in INTC_PSR_n are zero will not cause a preemption because their PRI_n will not be higher than PRI in the associated INTC_CPR_n.

Another function of the comparator is to signal an update of the INTC_IACKR_n with the vector number of the first interrupt that arrives that has a priority higher than the current priority. Once the vector number and priority are captured, they cannot be superseded by a higher priority interrupt until an update of the INTC_CPR_n occurs.

15.8.2.2 Stack (LIFO)

The LIFO stores the preempted PRI values from the associated `INTC_CPRn`. Therefore, because these priorities are stacked within the INTC, if interrupts need to be enabled during the ISR, at the beginning of the interrupt exception handler the PRI value in the associated `INTC_CPRn` does not need to be loaded from the associated `INTC_CPRn` and stored onto the context stack. Likewise at the end of the interrupt exception handler, the priority does not need to be loaded from the context stack and stored into the associated `INTC_CPRn`.

The PRI value in the associated `INTC_CPRn` is pushed onto the LIFO when the associated `INTC_IACKRn` is read in software vector mode or the interrupt acknowledge signal from the associated processor is asserted in hardware vector mode. The priority is popped into PRI in the associated `INTC_CPRn` whenever the associated `INTC_EOIRn` is written. An exception case in hardware vector mode to this behavior is described in [Hardware vector mode](#).

Although the INTC supports 16 priorities, an ISR executing with PRI in the `INTC_CPRn` equal to 15 will not be preempted. Therefore, the LIFO supports the stacking of 15 priorities. However, the LIFO is only 14 entries deep. An entry for a priority of 0 is not needed because of the ways that pushing onto a full LIFO and popping an empty LIFO are treated. If the LIFO is pushed 15 or more times than it is popped, the priorities first pushed are overwritten. A priority of 0 would be an overwritten priority. However, the LIFO will pop zeroes if it is popped more times than it is pushed. Therefore, although a priority of 0 was overwritten, it is regenerated with the popping of an empty LIFO.

15.8.3 Handshaking with processor

This section describes handshaking with the processor.

15.8.3.1 Software vector mode handshaking

This section describes the software vector mode handshaking.

15.8.3.1.1 Acknowledging interrupt request to processor

A timing diagram of the interrupt request and acknowledge handshaking in software vector mode, along with the handshaking near the end of the interrupt exception handler, is shown in [Figure 15-539](#). The INTC examines the peripheral and software-settable interrupt requests. When it finds an asserted peripheral or software-settable interrupt request with a higher priority than PRI in the associated `INTC_CPRn`, it asserts the

interrupt request to the associated processor. The INTVEC field in the associated INTC_IACKR n is updated with the preempting interrupt request's vector when the interrupt request to the processor is asserted. The INTVEC field retains that value until the next time the interrupt request to the processor is asserted. The rest of the handshaking is described in [Software vector mode](#).

15.8.3.1.2 End of interrupt exception handler

Before the interrupt exception handling completes, INTC_EOIR_PRC n must be written. When it is written, the associated LIFO is popped so that the preempted priority is restored into PRI of the associated INTC_CPR_PRC n . Before it is written, the peripheral or software-settable flag bit must be cleared so that the peripheral or software-settable interrupt request is negated.

NOTE

To ensure proper operation across all Power Architecture MCUs, execute an **mbar** or **msync** instruction between the access to clear the flag bit and the write to the INTC_EOIR_PRC n .

When returning from the preemption, the INTC does not search for the peripheral or software-settable interrupt request whose ISR was preempted. Depending on how much the ISR has progressed, that interrupt request may no longer even be asserted. When PRI in the associated INTC_CPR_PRC n is lowered to the priority of the preempted ISR, the interrupt request for the preempted ISR or any other asserted peripheral or software-settable interrupt request at or below that priority will not cause a preemption. Instead, after the restoration of the preempted context, the processor will return to the instruction address that it had been going to execute next, before it was preempted. This next instruction is part of the preempted ISR or the interrupt exception handler's prolog or epilog.

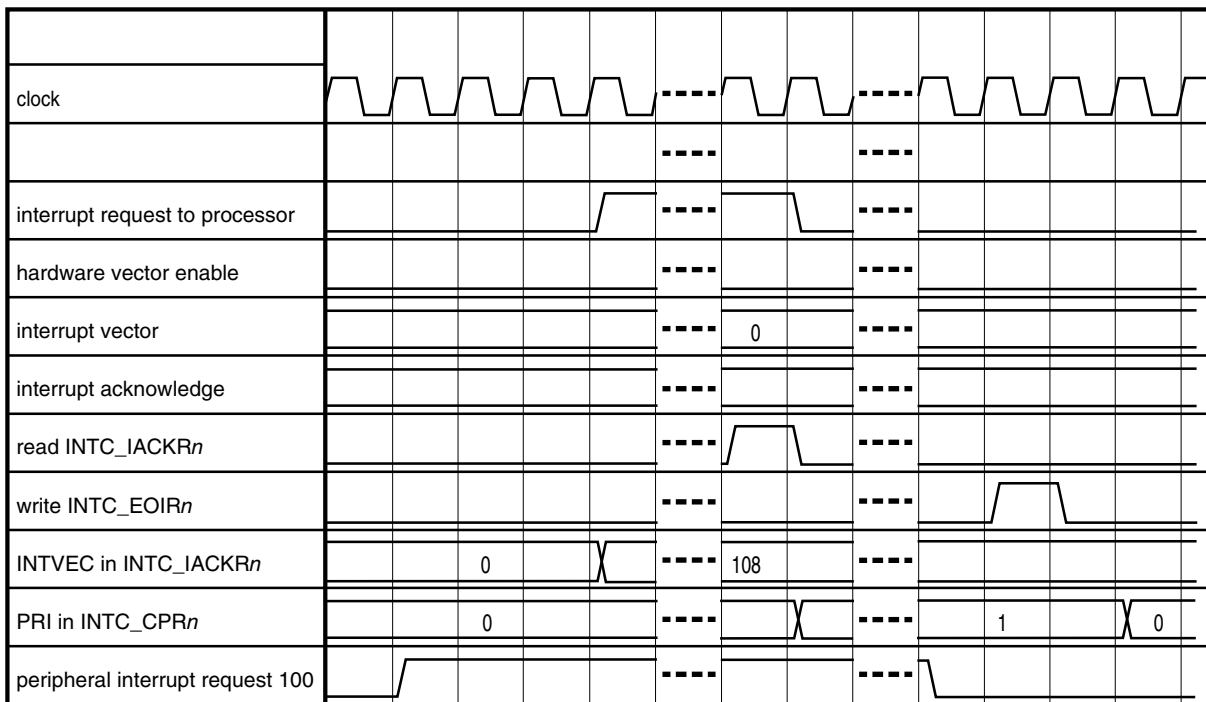


Figure 15-539. Timing diagram of software vector mode handshaking

15.8.3.2 Hardware vector mode handshaking

A timing diagram of the interrupt request and acknowledge handshaking in hardware vector mode, along with the handshaking near the end of the interrupt exception handler, is shown in [Figure 15-540](#). As in software vector mode, the INTC examines the peripheral and software-settable interrupt requests, and when it finds an asserted interrupt request with a higher priority than PRI in the associated INTC_CPR n , it asserts the interrupt request to the associated processor. The INTVEC field in the associated INTC_IACKR n is updated with the preempting peripheral or software-settable interrupt request's vector when the interrupt request to the processor is asserted. The INTVEC field retains that value until the next time the interrupt request to the associated processor is asserted. In addition, the value of the interrupt vector to the associated processor matches the value of the INTVEC field in the associated INTC_IACKR n . The rest of the handshaking is described in [Hardware vector mode](#).

The handshaking near the end of the interrupt exception handler, that is the writing to the associated INTC_EOIR n , is the same as in software vector mode. See [End of interrupt exception handler](#).

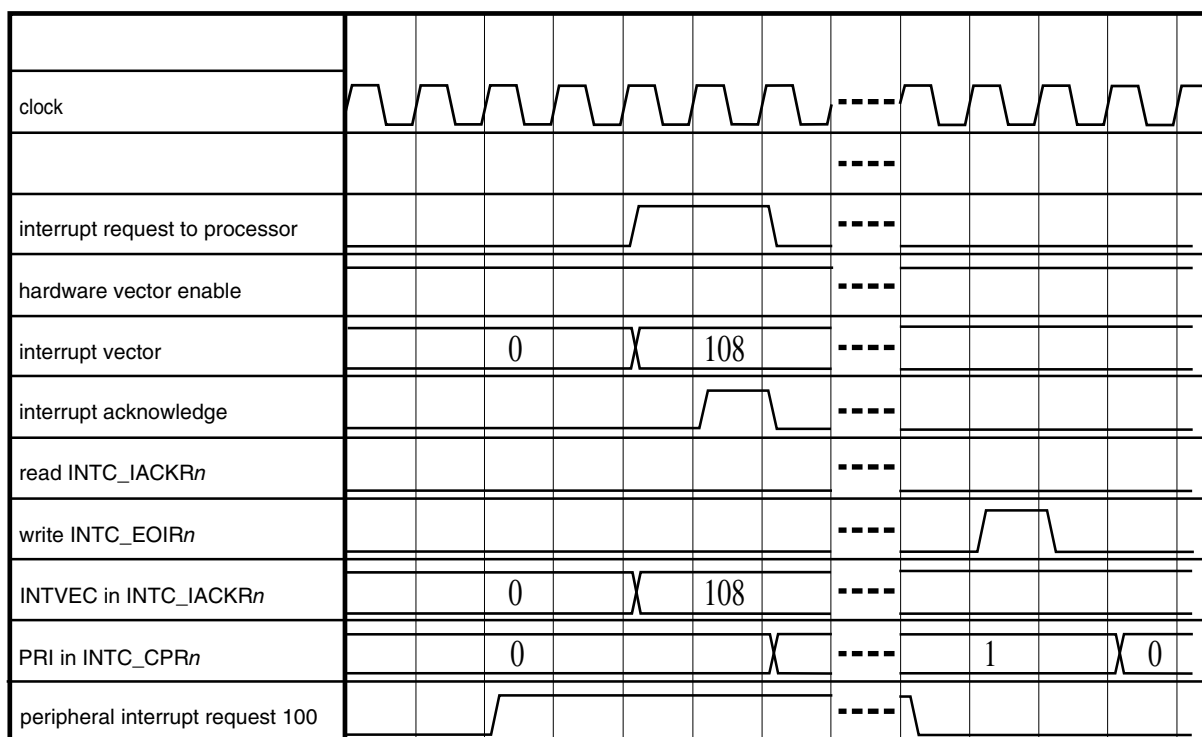


Figure 15-540. Timing diagram for hardware vector mode handshaking

15.9 Initialization/application information

This section contains initialization/application information.

15.9.1 Initialization flow

After exiting reset, all of the PRI_n and PRC_SEL_n fields in $INTC\_PSR_n$ are zero, and PRI in the $INTC\_CPR_n$ registers is 15d. These reset values prevent the INTC from asserting the interrupt request to the processors. The enable or mask bits in the peripherals are reset such that the peripheral interrupt requests are negated. An initialization sequence for allowing the peripheral and software-settable interrupt requests to cause an interrupt request to the processor is:

- interrupt_request_initialization:
 - Configure $HVEN\_PRC_n$ in $INTC\_MCR$.
 - Configure $VTBA\_PRC_n$ in $INTC\_IACKR\_PRC_n$.
 - Raise the PRI_n fields and set the PRC_SEL_n fields to the desired processor in $INTC\_PSR_n$.
 - Set the enable bits or clear the mask bits for the peripheral interrupt requests.

- Lower PRI in INTC_CPR_PRCn to zero.
- Enable recognition of interrupts by the processor(s).

15.9.2 Interrupt exception handler

These example interrupt exception handlers excerpts use Power Architecture assembly code.

15.9.2.1 Software vector mode

```

interrupt_exception_handler:
b        interrupt_exception_handler_continued# 16 bytes available, branch to continue

interrupt_exception_handler_continued:
code to create stack frame, save working register, and save SRR0 and SRR1 (Power
Architecture Save/Restore registers) (not shown)

lis      r3,INTC_IACKR_PRCn@ha      # form adjusted upper half of INTC_IACKR_PRCn address
lwz      r3,INTC_IACKR_PRCn@l(r3)   # load INTC_IACKR_PRCn, which clears request to
processor
lwz      r3,0x0(r3)                 # load address of ISR from vector table
wrteei   1                          # enable processor recognition of interrupts

code to save rest of context required by Power Architecture EABI (Embedded Binary
Application Interface) (not shown)

mtlrr    r3                        # move the INTC_IACKRn address into the link register
blrl     # branch to ISR; link register updated with epilog
# address

epilog:
code to restore most of context required by Power Architecture EABI (not shown)
# Popping the LIFO after the restoration of most of the context and the disabling of
processor
# recognition of interrupts eases the calculation of the maximum stack depth at the cost of
# postponing the servicing of the next interrupt request.
mbar     # ensure store to clear flag bit has completed
lis      r3,INTC_EOIR_PRCn@ha      # form adjusted upper half of INTC_EOIR_PRCn address
li       r4,0x0                    # form 0 to write to INTC_EOIR_PRCn
wrteei   0                          # disable processor recognition of interrupts
stw      r4,INTC_EOIR_PRCn@l(r3)   # store to INTC_EOIR_PRCn, informing INTC to lower
priority
code to restore SRR0 and SRR1, restore working registers, and delete stack frame (not shown)

rfi

vector_table_base_address:
address of ISR for interrupt with vector 0
address of ISR for interrupt with vector 1
.
.
.
address of ISR for interrupt with vector 510
address of ISR for interrupt with vector 511
ISRx:
code to service the interrupt event (not shown)
code to clear flag bit which drives interrupt request to INTC (not shown)
blr     # return to epilog

```

15.9.2.2 Hardware vector mode

This interrupt exception handler is useful with processor and system bus implementations that support a hardware vector. This example assumes that each

`interrupt_exception_handlerx` only has space for four instructions, and therefore a branch to `interrupt_exception_handler_continuedx` is needed.

```

interrupt_exception_handlerx:
b   interrupt_exception_handler_continuedx# 16 bytes available, branch to continue

interrupt_exception_handler_continuedx:
code to create stack frame, save working register, and save SRR0 and SRR1 (not shown)

wrteei 1      # enable processor recognition of interrupts

code to save rest of context required by Power Architecture EABI (not shown)

bl  ISRx      # branch to ISR for interrupt with vector x

epilog:
code to restore most of context required by Power Architecture EABI (not shown)

# Popping the LIFO after the restoration of most of the context and the disabling of
processor
# recognition of interrupts eases the calculation of the maximum stack depth at the cost of
# postponing the servicing of the next interrupt request.
mbar          # ensure store to clear flag bit has completed
lis  r3,INTC_EOIR_PRCn@ha # form adjusted upper half of INTC_EOIR_PRCn address
li   r4,0x0    # form 0 to write to INTC_EOIR_PRCn
wrteei 0      # disable processor recognition of interrupts
stw  r4,INTC_EOIR_PRCn@l(r3) # store to INTC_EOIR_PRCn, informing INTC to lower priority

code to restore SRR0 and SRR1, restore working registers, and delete stack frame (not shown)

rfi

ISRx:
code to service the interrupt event
code to clear flag bit which drives interrupt request to INTC

blr # branch to epilog

```

15.9.3 ISR, RTOS, and task hierarchy

The RTOS and all of the tasks under its control typically execute with PRI in `INTC_CPRn` having a value of 0. The RTOS will execute the tasks according to whatever priority scheme it may have, but that priority scheme is independent and has a lower priority of execution than the priority scheme of the INTC. In other words, the ISRs execute above `INTC_CPRn` priority 0 and outside the control of the RTOS, the RTOS executes at `INTC_CPRn` priority 0, and while the tasks execute at different priorities under the control of the RTOS, they also execute at `INTC_CPRn` priority 0.

If a task shares a resource with an ISR and the Priority Ceiling Protocol (PCP) is being used to manage that shared resource, then the task's priority can be elevated in the `INTC_CPRn` while the shared resource is being accessed.

An ISR whose `PRIn` in `INTC_PSRn` has a value of 0 will not cause an interrupt request to the selected processor, even if its peripheral or software-settable interrupt request is asserted. For a peripheral interrupt request, not setting its enable bit or disabling the mask bit will cause it to remain negated, which consequently also will not cause an interrupt request to the processor. Since the ISRs are outside the control of the RTOS, this ISR will not run unless called by another ISR or the interrupt exception handler, perhaps after executing another ISR.

15.9.4 Order of execution

An ISR with a higher priority can preempt an ISR with a lower priority, regardless of the unique vectors associated with each of their peripheral or software-settable interrupt requests. However, if multiple peripheral or software-settable interrupt requests are asserted, more than one of these interrupt requests has the highest priority and that priority is high enough to cause preemption, the INTC selects the one with the lowest unique vector regardless of the order in time that they asserted. However, the ability to meet deadlines with this scheduling scheme is no less than if the ISRs execute in the time order that their peripheral or software-settable interrupt requests asserted.

The example in the following table shows the order of execution of both cases, ISRs with different priorities and ISRs with the same priority.

Table 15-534. Order of ISR execution example

Step #	Step Description	Code executing at end of step						PRI in INTC_CPR at end of step
		RTOS	ISR108 ₁	ISR208	ISR308	ISR408	interrupt exception handler	
1	RTOS at priority 0 is executing.	X						0
2	Peripheral interrupt request 100 at priority 1 asserts. Interrupt taken.		X					1
3	Peripheral interrupt request 400 at priority 4 asserts. Interrupt taken.					X		4
4	Peripheral interrupt request 300 at priority 3 asserts.					X		4
5	Peripheral interrupt request 200 at priority 3 asserts.					X		4
6	ISR408 completes. Interrupt exception handler writes to <code>INTC_EOIR_n</code> .						X	1

Table continues on the next page...

Table 15-534. Order of ISR execution example (continued)

Step #	Step Description	Code executing at end of step						PRI in INTC_CPR at end of step
		RTOS	ISR108 ¹	ISR208	ISR308	ISR408	interrupt exception handler	
7	Interrupt taken. ISR208 starts to execute, even though peripheral interrupt request 300 asserted first.			X				3
8	ISR208 completes. Interrupt exception handler writes to INTC_EOIR _n .						X	1
9	Interrupt taken. ISR308 starts to execute.				X			3
10	ISR308 completes. Interrupt exception handler writes to INTC_EOIR _n .						X	1
11	ISR108 completes. Interrupt exception handler writes to INTC_EOIR _n .						X	0
12	RTOS continues execution.	X						0

1. ISR108 executes for peripheral interrupt request 100 because the first eight ISRs are for software-settable interrupt requests.

15.9.5 Priority ceiling protocol

This section describes the priority ceiling protocol.

15.9.5.1 Elevating priority

The PRI field in INTC current priority register (INTC_CPR_n) is elevated in the OSEK PCP to the ceiling of all of the priorities of the ISRs that share a resource. This protocol therefore allows coherent accesses of the ISRs to that shared resource.

For example, ISR1 has a priority of 1, ISR2 has a priority of 2, and ISR3 has a priority of 3. They all share the same resource. Before ISR1 or ISR2 can access that resource, they must raise the PRI value in INTC_CPR_n to 3, the ceiling of all of the ISR priorities. After they release the resource, they must lower the PRI value in INTC_CPR_n to prevent further priority inversion. If they do not raise their priority, then ISR2 can preempt ISR1, and ISR3 can preempt ISR1 or ISR2, possibly corrupting the shared resource. Another possible failure mechanism is deadlock if the higher priority ISR needs the lower priority ISR to release the resource before it can continue, but the lower priority ISR cannot release the resource until the higher priority ISR completes and execution returns to the lower priority ISR.

Using the PCP instead of disabling processor recognition of all interrupts eliminates the time when accessing a shared resource that all higher priority interrupts are blocked. For example, while ISR3 cannot preempt ISR1 while it is accessing the shared resource, all of the ISRs with a priority higher than 3 can preempt ISR1.

15.9.5.2 Ensuring coherency

This section discusses how to ensure coherency.

15.9.5.2.1 Interrupt with blocked priority

A scenario can exist that can cause non-coherent accesses to the shared resource. As an example, ISR1 and ISR2 both share a resource. ISR1 has a lower priority than ISR2. ISR1 is executing, and it writes to the `INTC_CPRn`. The instruction following this store is a store to a value in a shared coherent data block. Either just before or at the same time as the first store, the INTC asserts the interrupt request to the processor because the peripheral interrupt request for ISR2 has asserted. As the processor is responding to the interrupt request from the INTC, and as it is aborting transactions and flushing its pipeline, it is possible (in some implementations) that both of these stores are executed. ISR2 thereby assumes that it can access the data block coherently, but the data block has been corrupted.

When the Current Priority Register's (INTC_CPR_PCR's) PRI field in the INTC is updated by software, external interrupts from the INTC to the core can be disabled so that an external interrupt will never be taken in the window of time when a system resource (PRI field) is being changed. Disabling interrupts and re-enabling them after the PRI elevation, as well as providing additional core clocks of delay after interrupts have been enabled, provides enough delay for the ISR that has elevated its PRI to be interrupted by a pending ISR before the ISR elevating PRI accesses a shared memory block.

To ensure coherent access to the shared data block, modifications to PRI in INTC_CPR_PRCx can be made by services managing shared resources with the following code sequence:

- disable processor recognition of interrupts
- PRI modification (guarded, cache inhibited)
- enable processor recognition of interrupts
- delay 5 core clocks
- coherent data access

The delay of 5 clocks minimum can be achieved by the following series of instructions:

```
or r1, r1, r1
```



```

or r1, r1, r1
or r1, r1, r1
or r1, r1, r1
or r1, r1, r1

```

The `or r1, r1, r1` is not dual issued because the destination of one “or” instruction is the source of the next, hence the instructions cannot be executed in parallel.

15.9.5.2.2 Raised priority preserved

The following figure shows the timing diagram for a scenario where a priority that is raised in one ISR is preserved after execution of a pre-empting ISR. The table following the figure explains the events. The example is for software vector mode. Except for the method of retrieving the vector and acknowledging the interrupt request to the processor, hardware vector mode is identical.

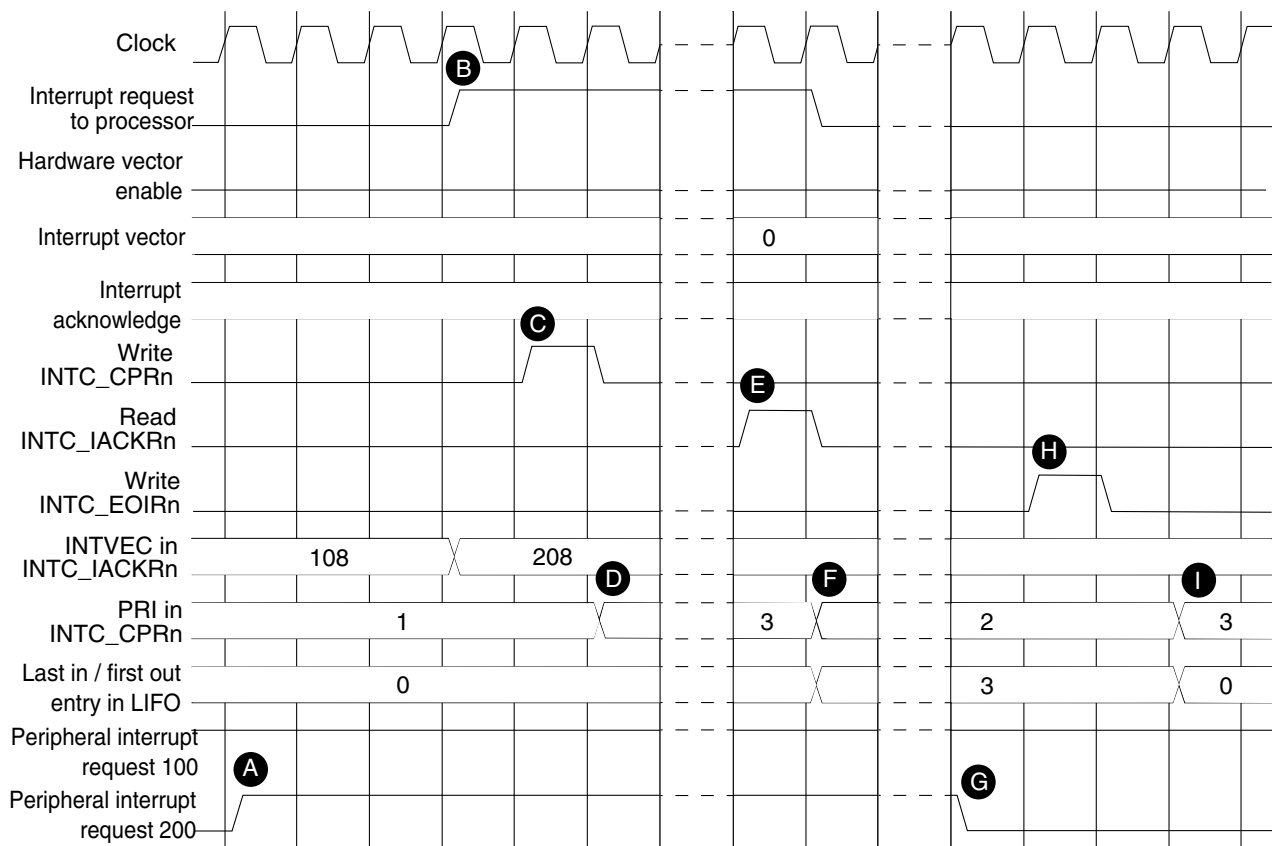


Figure 15-541. Timing diagram of raised priority preserved

Table 15-535. Raised priority preserved events

Event	Description
A	Peripheral interrupt request 200 asserts during execution of ISR108 running at priority 1.
B	Interrupt request to processor asserts. INTVEC in INTC_IACKRn updates with vector for that peripheral interrupt request.

Table continues on the next page...

Table 15-535. Raised priority preserved events (continued)

Event	Description
C	ISR108 writes to INTC_CPR _n to raise priority to 3 before accessing shared coherent data block.
D	PRI in INTC_CPR _n now at 3, reflecting the write. This write, just before accessing data block, is the last instruction the processor executes before being interrupted.
E	Interrupt exception handler prolog acknowledges interrupt by reading INTC_IACKR _n .
F	PRI of 3 pushed onto LIFO. PRI in INTC_CPR _n updates to 2, the priority of ISR208.
G	ISR208 clears its flag bit, deasserting its peripheral interrupt request.
H	Interrupt exception handler epilog writes to INTC_EOIR _n .
I	LIFO pops 3, restoring the raised priority onto PRI in INTC_CPR _n . Next value to pop from LIFO is the priority from before peripheral interrupt request 100 interrupted. ISR108 now can access data block coherently after interrupt exception handler executes rfi instruction.

15.9.6 Selecting priorities according to request rates and deadlines

The selection of the priorities for the ISRs can be made using Rate Monotonic Scheduling (RMS) or a superset of it, Deadline Monotonic Scheduling (DMS). In RMS, the ISRs which have higher request rates have higher priorities. In DMS, if the deadline is before the next time the ISR is requested, then the ISR is assigned a priority according to the time from the request for the ISR to the deadline, not from the time of the request for the ISR to the next request for it.

For example, ISR1 executes every 100 μ s, ISR2 executes every 200 μ s, and ISR3 executes every 300 μ s. ISR1 has a higher priority than ISR2 which has a higher priority than ISR3. However, if ISR3 has a deadline of 150 μ s, then it has a higher priority than ISR2.

The INTC supports 16 priorities, which may be many fewer than the number of ISRs. In this case, the ISRs should be grouped with other ISRs that have similar deadlines. For example, a priority could be allocated for every time the request rate doubles. ISRs with request rates around 1 ms would share a priority, ISRs with request rates around 500 μ s would share a priority, ISRs with request rates around 250 μ s would share a priority, and so on. With this approach, a range of ISR request rates of 2^{16} could be covered, regardless of the number of ISRs.

Reducing the number of priorities does cause some priority inversion, which reduces the processor's ability to meet its deadlines. However, reducing the number of priorities can reduce the size and latency through the interrupt controller. It also allows easier management of ISRs with similar deadlines that share a resource. They can be placed at the same priority without any further priority inversion, and they do not need to use the PCP to access the shared resource.

15.9.7 Software-settable interrupt requests

The software-settable interrupt requests can be used in two ways. They can be used to schedule a lower priority portion of an ISR and for processors to interrupt other processors in a multiple processor system.

15.9.7.1 Scheduling a lower priority portion of an ISR

A portion of an ISR needs to be executed at the PRI_n value in $INTC\_PSR_n$, which becomes the PRI value in $INTC\_CPR_n$ with the interrupt acknowledge. The ISR, however, can have a portion of it which does not need to be executed at this higher priority. Therefore, executing this later portion which does not need to be executed at this higher priority can prevent the execution of ISRs which do not have a higher priority than the earlier portion of the ISR but do have a higher priority than the later portion of the ISR needs. This preemptive scheduling inefficiency reduces the processor's ability to meet its deadlines.

One option is for the ISR to complete the earlier higher priority portion, but then schedule through the RTOS a task to execute the later lower priority portion. However, some RTOSs can require a large amount of time for an ISR to schedule a task. Therefore, a second option is for the ISR, after completing the higher priority portion, to set a SET_n bit in $INTC\_SSCIR_n$. Writing a '1' to SET_n causes a software-settable interrupt request. This software-settable interrupt request, which usually will have a lower PRI_n value in the $INTC\_PSR_n$, therefore will not cause preemptive scheduling inefficiencies.

After generating a software settable interrupt request, the higher priority ISR completes. The lower priority ISR is scheduled according to its priority. Execution of the higher priority ISR is not resumed after the completion of the lower priority ISR.

15.9.7.2 Scheduling an ISR on another processor

Since the SET_n bits in the $INTC\_SSCIR_n$ are memory mapped, processors in multiple processor systems can schedule ISRs on the other processors. One possible application is if one processor simply wants to command another processor to perform a piece of work, and the initiating processor does not need to use the results of that work. If the initiating processor needs to know if the processor executing the software-settable ISR has not completed the work before asking it to again execute that ISR, it can check if the corresponding CLR_n bit in $INTC\_SSCIR_n$ is asserted before again writing a 1 to the SET_n bit.

Another application is the sharing of a block of data. For example, a first processor has completed accessing a block of data and wants a second processor to then access it. Furthermore, after the second processor has completed accessing the block of data, the first processor again wants to access it. The accesses to the block of data must be done coherently. The procedure is that the first processor writes a 1 to a SET n bit on the second processor. The second processor, after accessing the block of data, clears the corresponding CLR n bit and then writes 1 to a SET n bit on the first processor, informing it that it now can access the block of data.

15.9.8 Lowering priority within an ISR

In implementations without the software-settable interrupt requests in INTC_SSCIR n , one way (besides scheduling a task through an RTOS) to prevent preemptive scheduling inefficiencies with an ISR whose work spans multiple priorities (as described in [Scheduling a lower priority portion of an ISR](#)) is to lower the current priority. However, the INTC has a LIFO whose depth is determined by the number of priorities.

Note

Lowering the PRI value in INTC_CPR n within an ISR to below the ISR's corresponding PRI value in INTC_PSR n allows more preemptions than the depth of the LIFO can support.

Therefore, through its use of the LIFO the INTC does not support lowering the current priority within an ISR as a way to avoid preemptive scheduling inefficiencies.

15.9.9 Negating an interrupt request outside of its ISR

This section discusses the negation of an interrupt service request outside of its ISR.

15.9.9.1 Negating an interrupt request as a side effect of an ISR

Some peripherals have flag bits which can be cleared as a side effect of servicing a peripheral interrupt request. For example, reading a specific register can clear the flag bits, and consequently their corresponding interrupt requests too. This clearing as a side effect of servicing a peripheral interrupt request can cause the negation of other peripheral interrupt requests besides the peripheral interrupt request whose ISR presently is executing. This negating of a peripheral interrupt request outside of its ISR can be a desired effect.

15.9.9.2 Negating multiple interrupt requests in one ISR

An ISR can clear other flag bits besides its own flag bit. One reason that an ISR clears multiple flag bits is because it serviced those other flag bits, and therefore the ISRs for these other flag bits do not need to be executed.

15.9.9.3 Proper setting of interrupt request priority

Whether an interrupt request negates outside of its own ISR due to the side effect of an ISR execution or the intentional clearing of a flag bit, the priorities of the peripheral or software-settable interrupt requests for these other flag bits must be selected properly. Their PRI_n values in $INTC\_PSR_n$ must be selected to be at or lower than the priority of the ISR that cleared their flag bits. Otherwise, those flag bits still can cause the interrupt request to the processor to assert. Furthermore, the clearing of these other flag bits also has the same timing relationship to the writing to $INTC\_SSCIR_n$ as the clearing of the flag bit that caused the present ISR to be executed. Refer to [End of interrupt exception handler](#) for more information.

A flag bit whose enable bit or mask bit is negating its peripheral interrupt request can be cleared at any time, regardless of the peripheral interrupt request's PRI_n value in $INTC\_PSR_n$.

15.9.10 Examining LIFO contents

Normally the user does not need to know the contents of the LIFO, or even how deep the LIFO is nested. Although the LIFO contents are not memory-mapped, the user can read the contents by popping the LIFO and reading the PRI field in the INTC current priority register ($INTC\_CPR_n$). Disabling processor recognition of interrupts while examining the LIFO contents provides a coherent view of the preempted priorities. The code sequence is:

```
pop_lifo:
    store to INTC_EOIR_PRCn
    load INTC_CPR_PRCn, examine PRI, and store onto stack
    if PRI is not zero or value when interrupts were enabled, branch to pop_lifo
```

When the examination is complete, the LIFO can be restored in software vector mode using this code sequence:

```
push_lifo:
    load stacked PRI value and store to INTC_CPR_PRCn
```

```
load INTC_IACKR_PRCn  
if stacked PRI values are not depleted, branch to push_lifo
```

Note

Reading the INTC_IACKR_PRC n acknowledges the interrupt request to the processor and updates the INTC_CPR_PRC n [PRI] with the priority of the preempting interrupt request. If the processor recognition of interrupts is disabled during the LIFO restoration, interrupt requests to the processor can go undetected. However, because the peripheral or software-settable interrupt requests are not cleared, the peripheral interrupt request to the processor re-asserts when INTC_CPR_PRC n [PRI] is lower than the priorities of those peripheral or software-settable interrupt requests.

15.10 Interrupt sources

See [Interrupt vector assignments](#) for the chip-specific list of interrupt sources.

Chapter 16

Direct Memory Access Controller (eDMA)

16.1 Chip-specific eDMA information

The chip has two instances of the eDMA module: eDMA_A and eDMA_B. Each instance supports 64 channels.

See [DMA request mapping](#) for the channel allocation of DMA requests for each module instance.

The eDMA has no direct external MCU signals. However, four external pins can be configured in the SIU to route external eDMA requests to four eDMA_A channels. See [DMA/Interrupt Request Select Register \(SIU_DIRSR\)](#) for details.

16.2 Introduction

The enhanced direct memory access (eDMA) controller is a second-generation module capable of performing complex data transfers with minimal intervention from a host processor. The hardware microarchitecture includes:

- A DMA engine that performs:
 - Source address and destination address calculations
 - Data-movement operations
- Local memory containing transfer control descriptors for each of the 64 channels

16.2.1 eDMA system block diagram

[Figure 16-1](#) illustrates the components of the eDMA system, including the eDMA module ("engine").

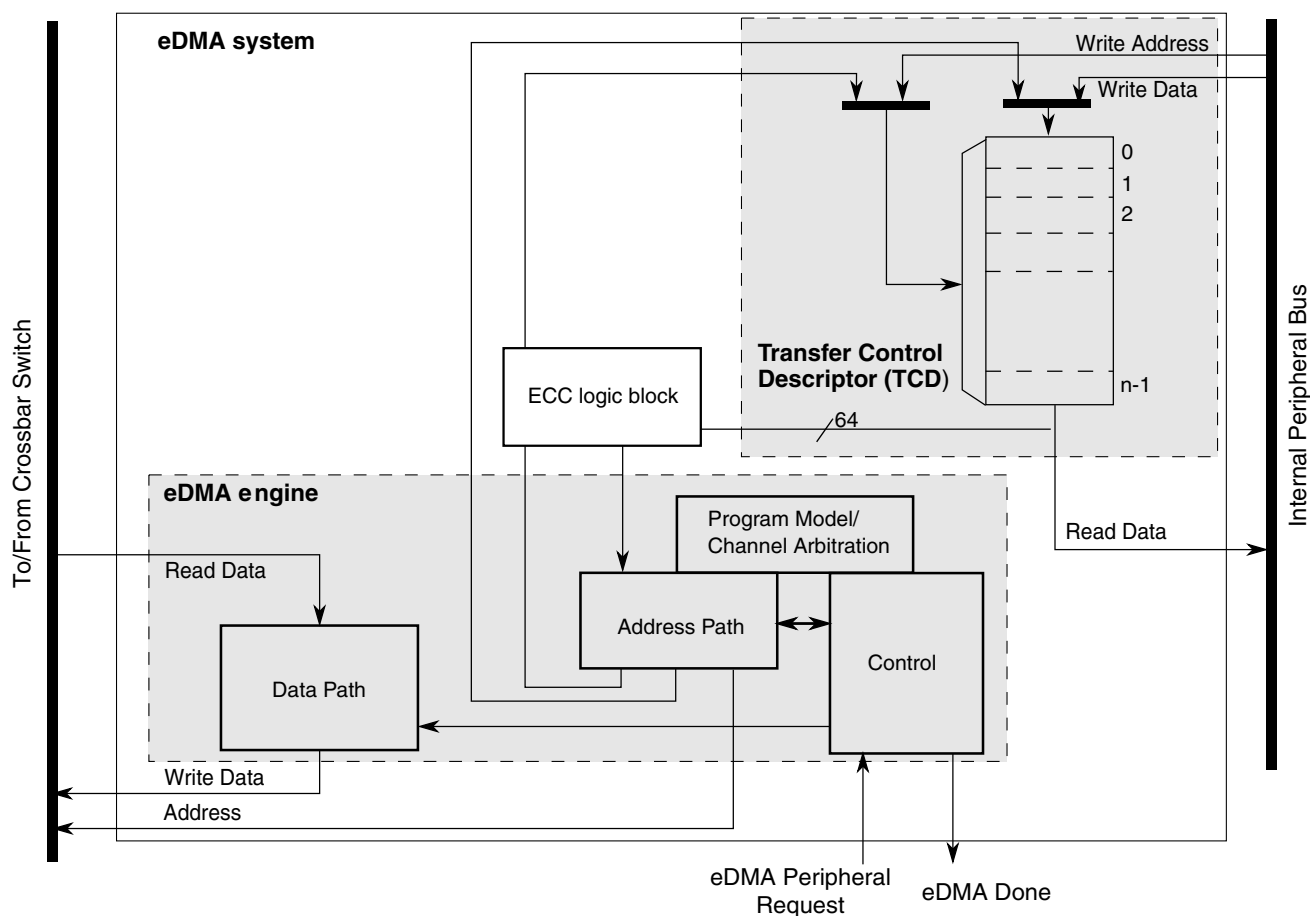


Figure 16-1. eDMA system block diagram

16.2.2 Block parts

The eDMA module is partitioned into two major modules: the eDMA engine and the transfer-control descriptor local memory.

The eDMA engine is further partitioned into four submodules:

Table 16-1. eDMA engine submodules

Submodule	Function
Address path	This block implements registered versions of two channel transfer control descriptors, channel x and channel y, and manages all master bus-address calculations. All the channels provide the same functionality. This structure allows data transfers associated with one channel to be preempted after the completion of a read/write sequence if a higher priority channel activation is asserted while the first channel is active. After a channel is activated, it runs until the minor loop is completed, unless preempted by a higher priority channel. This provides a mechanism (enabled by DCHPRI_n[ECP]) where a large data move operation can be preempted to minimize the time another channel is blocked from execution. When any channel is selected to execute, the contents of its TCD are read from local memory and loaded into the address path channel x registers for a normal start and into channel y registers for a preemption start. After the minor loop completes execution, the address path hardware writes the new values for the TCD_n{SADDR, DADDR, CITER} back to local memory. If the major iteration count is exhausted, additional processing is performed, including the final address pointer updates, reloading the TCD_n_CITER field, and a possible fetch of the next TCD_n from memory as part of a scatter/gather operation.
Data path	This block implements the bus master read/write datapath. It includes a data buffer and the necessary multiplex logic to support any required data alignment. The internal read data bus is the primary input, and the internal write data bus is the primary output. The address and data path modules directly support the 2-stage pipelined internal bus. The address path module represents the 1st stage of the bus pipeline (address phase), while the data path module implements the 2nd stage of the pipeline (data phase).
Program model/channel arbitration	This block implements the first section of the eDMA programming model as well as the channel arbitration logic. The programming model registers are connected to the internal peripheral bus. The eDMA peripheral request inputs and interrupt request outputs are also connected to this block (via control logic).
Control	This block provides all the control functions for the eDMA engine. For data transfers where the source and destination sizes are equal, the eDMA engine performs a series of source read/destination write operations until the number of bytes specified in the minor loop byte count has moved. For descriptors where the sizes are not equal, multiple accesses of the smaller size data are required for each reference of the larger size. As an example, if the source size references 16-bit data and the destination is 32-bit data, two reads are performed, then one 32-bit write.

The transfer-control descriptor local memory is further partitioned into:

Table 16-2. Transfer control descriptor memory

Submodule	Description
Memory controller	This logic implements the required dual-ported controller, managing accesses from the eDMA engine as well as references from the internal peripheral bus. As noted earlier, in the event of simultaneous accesses, the eDMA engine is given priority and the peripheral transaction is stalled.
Memory array	TCD storage for each channel's transfer profile.

16.2.3 Features

The eDMA is a highly programmable data-transfer engine optimized to minimize any required intervention from the host processor. It is intended for use in applications where the data size to be transferred is statically known and not defined within the transferred data itself. The eDMA module features:

- All data movement via dual-address transfers: read from source, write to destination
 - Programmable source and destination addresses and transfer size
 - Support for enhanced addressing modes
- 64-channel implementation that performs complex data transfers with minimal intervention from a host processor
 - Internal data buffer, used as temporary storage to support 16- and 32-byte transfers
 - Connections to the crossbar switch for bus mastering the data movement
- Transfer control descriptor (TCD) organized to support two-deep, nested transfer operations
 - 32-byte TCD stored in local memory for each channel
 - An inner data transfer loop defined by a minor byte transfer count
 - An outer data transfer loop defined by a major iteration count
- Channel activation via one of three methods:
 - Explicit software initiation
 - Initiation via a channel-to-channel linking mechanism for continuous transfers
 - Peripheral-paced hardware requests, one per channel
- Fixed-priority and round-robin channel arbitration
- Channel completion reported via programmable interrupt requests
 - One interrupt per channel, which can be asserted at completion of major iteration count
 - Programmable error terminations per channel and logically summed together to form one error interrupt to the interrupt controller
- Programmable support for scatter/gather DMA processing

- Support for complex data structures
- Error detection and error correction

In the discussion of this module, n is used to reference the channel number.

16.3 Modes of operation

The eDMA operates in the following modes:

Table 16-3. Modes of operation

Mode	Description
Normal	In Normal mode, the eDMA transfers data between a source and a destination. The source and destination can be a memory block or an I/O block capable of operation with the eDMA. A service request initiates a transfer of a specific number of bytes (NBYTES) as specified in the transfer control descriptor (TCD). The minor loop is the sequence of read-write operations that transfers these NBYTES per service request. Each service request executes one iteration of the major loop, which transfers NBYTES of data.
Debug	DMA operation is configurable in Debug mode via the control register: • If CR[EDBG] is cleared, the DMA continues to operate. • If CR[EDBG] is set, the eDMA stops transferring data. If Debug mode is entered while a channel is active, the eDMA continues operation until the channel retires.
Wait	Before entering Wait mode, the DMA attempts to complete its current transfer. After the transfer completes, the device enters Wait mode.

16.4 Memory map/register definition

The eDMA's programming model is partitioned into two regions:

- The first region defines a number of registers providing control functions
- The second region corresponds to the local transfer control descriptor (TCD) memory

16.4.1 TCD memory

Each channel requires a 32-byte transfer control descriptor for defining the desired data movement operation. The channel descriptors are stored in the local memory in sequential order: channel 0, channel 1, ... channel 63. Each TCD_n definition is presented as 11 registers of 16 or 32 bits.

Some registers are implemented as two 32-bit registers, and include an H and L suffix, signaling the high and low portions of the control function. This table provides a 32-bit view of the eDMA's memory map.

16.4.2 TCD initialization

The initialization of the TCD memory is performed by the eDMA controller after reset. The initialization runs for 256 eDMA clock cycles. All fields in the TCD memory are initialized to 0. All application read or write accesses to the TCD are delayed until the initialization is finished. Prior to activating a channel, you must initialize its TCD with the appropriate transfer profile.

16.4.3 TCD structure

DMA Basics: TCD Structure

- One DMA engine has a number of channels to react to DMA requests
- Each channel has its own TCD

Word Offset	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
0x1000	SADDR																															
0x1004	SMOD				SSIZE				DMOD				DSIZE				SOFF															
0x1008	NBYTES ¹																															
0x1008	SMLOE ¹	DMLOE ¹	MLOFF or NBYTES ¹																				NBYTES ¹									
0x100C	SLAST																															
0x1010	DADDR																															
0x1014	CITER.E_LINK	CITER or CITER.LINKCH						CITER								DOFF																
0x1018	DLAST_SGA																															
0x101C	BITER.E_LINK	BITER or BITER.LINKCH						BITER								BWC		MAJOR LINKCH						DONE	ACTIVE	MAJOR.E_LINK	E_SG	D_REQ	INT_HALF	INT_MAJ	START	
	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31

¹ The fields implemented in Word 2 depend on whether EDMA_CR[EMLM] bit is set to 0 or 1.

16.4.4 Reserved memory and bit fields

- Reading reserved bits in a register returns the value of zero.
- Writes to reserved bits in a register are ignored.
- Reading or writing a reserved memory location generates a bus error.

DMA memory map

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
0	Control Register (DMA_CR)	32	R/W	0000_E400h	16.4.5/660
4	Error Status Register (DMA_ES)	32	R	0000_0000h	16.4.6/663
8	Enable Request Register High (DMA_ERQH)	32	R/W	0000_0000h	16.4.7/666

Table continues on the next page...

DMA memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
C	Enable Request Register Low (DMA_ERQL)	32	R/W	0000_0000h	16.4.8/669
10	Enable Error Interrupt Register High (DMA_EEIH)	32	R/W	0000_0000h	16.4.9/673
14	Enable Error Interrupt Register Low (DMA_EEIL)	32	R/W	0000_0000h	16.4.10/676
18	Set Enable Request Register (DMA_SERQ)	8	W (always reads 0)	00h	16.4.11/680
19	Clear Enable Request Register (DMA_CERQ)	8	W (always reads 0)	00h	16.4.12/681
1A	Set Enable Error Interrupt Register (DMA_SEEI)	8	W (always reads 0)	00h	16.4.13/682
1B	Clear Enable Error Interrupt Register (DMA_CEEI)	8	W (always reads 0)	00h	16.4.14/683
1C	Clear Interrupt Request Register (DMA_CINT)	8	W (always reads 0)	00h	16.4.15/684
1D	Clear Error Register (DMA_CERR)	8	W (always reads 0)	00h	16.4.16/685
1E	Set START Bit Register (DMA_SSRT)	8	W (always reads 0)	00h	16.4.17/686
1F	Clear DONE Status Bit Register (DMA_CDNE)	8	W (always reads 0)	00h	16.4.18/687
20	Interrupt Request Register High (DMA_INTH)	32	R/W	0000_0000h	16.4.19/687
24	Interrupt Request Register Low (DMA_INTL)	32	R/W	0000_0000h	16.4.20/691
28	Error Register High (DMA_ERRH)	32	R/W	0000_0000h	16.4.21/695
2C	Error Register Low (DMA_ERRL)	32	R/W	0000_0000h	16.4.22/699
30	Hardware Request Status Register High (DMA_HRSH)	32	R	0000_0000h	16.4.23/702
34	Hardware Request Status Register Low (DMA_HRSL)	32	R	0000_0000h	16.4.24/706
38	Global PCU Output Register (DMA_GPORH)	32	R/W	0000_0000h	16.4.25/710
3C	Global PCU Output Register (DMA_GPORL)	32	R/W	0000_0000h	16.4.25/710
100	Channel n Priority Register (DMA_DCHPRI0)	8	R/W	See section	16.4.26/711

Table continues on the next page...

DMA memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
101	Channel n Priority Register (DMA_DCHPRI1)	8	R/W	See section	16.4.26/ 711
102	Channel n Priority Register (DMA_DCHPRI2)	8	R/W	See section	16.4.26/ 711
103	Channel n Priority Register (DMA_DCHPRI3)	8	R/W	See section	16.4.26/ 711
104	Channel n Priority Register (DMA_DCHPRI4)	8	R/W	See section	16.4.26/ 711
105	Channel n Priority Register (DMA_DCHPRI5)	8	R/W	See section	16.4.26/ 711
106	Channel n Priority Register (DMA_DCHPRI6)	8	R/W	See section	16.4.26/ 711
107	Channel n Priority Register (DMA_DCHPRI7)	8	R/W	See section	16.4.26/ 711
108	Channel n Priority Register (DMA_DCHPRI8)	8	R/W	See section	16.4.26/ 711
109	Channel n Priority Register (DMA_DCHPRI9)	8	R/W	See section	16.4.26/ 711
10A	Channel n Priority Register (DMA_DCHPRI10)	8	R/W	See section	16.4.26/ 711
10B	Channel n Priority Register (DMA_DCHPRI11)	8	R/W	See section	16.4.26/ 711
10C	Channel n Priority Register (DMA_DCHPRI12)	8	R/W	See section	16.4.26/ 711
10D	Channel n Priority Register (DMA_DCHPRI13)	8	R/W	See section	16.4.26/ 711
10E	Channel n Priority Register (DMA_DCHPRI14)	8	R/W	See section	16.4.26/ 711
10F	Channel n Priority Register (DMA_DCHPRI15)	8	R/W	See section	16.4.26/ 711
110	Channel n Priority Register (DMA_DCHPRI16)	8	R/W	See section	16.4.26/ 711
111	Channel n Priority Register (DMA_DCHPRI17)	8	R/W	See section	16.4.26/ 711
112	Channel n Priority Register (DMA_DCHPRI18)	8	R/W	See section	16.4.26/ 711
113	Channel n Priority Register (DMA_DCHPRI19)	8	R/W	See section	16.4.26/ 711
114	Channel n Priority Register (DMA_DCHPRI20)	8	R/W	See section	16.4.26/ 711
115	Channel n Priority Register (DMA_DCHPRI21)	8	R/W	See section	16.4.26/ 711
116	Channel n Priority Register (DMA_DCHPRI22)	8	R/W	See section	16.4.26/ 711

Table continues on the next page...

DMA memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
117	Channel n Priority Register (DMA_DCHPRI23)	8	R/W	See section	16.4.26/ 711
118	Channel n Priority Register (DMA_DCHPRI24)	8	R/W	See section	16.4.26/ 711
119	Channel n Priority Register (DMA_DCHPRI25)	8	R/W	See section	16.4.26/ 711
11A	Channel n Priority Register (DMA_DCHPRI26)	8	R/W	See section	16.4.26/ 711
11B	Channel n Priority Register (DMA_DCHPRI27)	8	R/W	See section	16.4.26/ 711
11C	Channel n Priority Register (DMA_DCHPRI28)	8	R/W	See section	16.4.26/ 711
11D	Channel n Priority Register (DMA_DCHPRI29)	8	R/W	See section	16.4.26/ 711
11E	Channel n Priority Register (DMA_DCHPRI30)	8	R/W	See section	16.4.26/ 711
11F	Channel n Priority Register (DMA_DCHPRI31)	8	R/W	See section	16.4.26/ 711
120	Channel n Priority Register (DMA_DCHPRI32)	8	R/W	See section	16.4.26/ 711
121	Channel n Priority Register (DMA_DCHPRI33)	8	R/W	See section	16.4.26/ 711
122	Channel n Priority Register (DMA_DCHPRI34)	8	R/W	See section	16.4.26/ 711
123	Channel n Priority Register (DMA_DCHPRI35)	8	R/W	See section	16.4.26/ 711
124	Channel n Priority Register (DMA_DCHPRI36)	8	R/W	See section	16.4.26/ 711
125	Channel n Priority Register (DMA_DCHPRI37)	8	R/W	See section	16.4.26/ 711
126	Channel n Priority Register (DMA_DCHPRI38)	8	R/W	See section	16.4.26/ 711
127	Channel n Priority Register (DMA_DCHPRI39)	8	R/W	See section	16.4.26/ 711
128	Channel n Priority Register (DMA_DCHPRI40)	8	R/W	See section	16.4.26/ 711
129	Channel n Priority Register (DMA_DCHPRI41)	8	R/W	See section	16.4.26/ 711
12A	Channel n Priority Register (DMA_DCHPRI42)	8	R/W	See section	16.4.26/ 711
12B	Channel n Priority Register (DMA_DCHPRI43)	8	R/W	See section	16.4.26/ 711
12C	Channel n Priority Register (DMA_DCHPRI44)	8	R/W	See section	16.4.26/ 711

Table continues on the next page...

DMA memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
12D	Channel n Priority Register (DMA_DCHPRI45)	8	R/W	See section	16.4.26/ 711
12E	Channel n Priority Register (DMA_DCHPRI46)	8	R/W	See section	16.4.26/ 711
12F	Channel n Priority Register (DMA_DCHPRI47)	8	R/W	See section	16.4.26/ 711
130	Channel n Priority Register (DMA_DCHPRI48)	8	R/W	See section	16.4.26/ 711
131	Channel n Priority Register (DMA_DCHPRI49)	8	R/W	See section	16.4.26/ 711
132	Channel n Priority Register (DMA_DCHPRI50)	8	R/W	See section	16.4.26/ 711
133	Channel n Priority Register (DMA_DCHPRI51)	8	R/W	See section	16.4.26/ 711
134	Channel n Priority Register (DMA_DCHPRI52)	8	R/W	See section	16.4.26/ 711
135	Channel n Priority Register (DMA_DCHPRI53)	8	R/W	See section	16.4.26/ 711
136	Channel n Priority Register (DMA_DCHPRI54)	8	R/W	See section	16.4.26/ 711
137	Channel n Priority Register (DMA_DCHPRI55)	8	R/W	See section	16.4.26/ 711
138	Channel n Priority Register (DMA_DCHPRI56)	8	R/W	See section	16.4.26/ 711
139	Channel n Priority Register (DMA_DCHPRI57)	8	R/W	See section	16.4.26/ 711
13A	Channel n Priority Register (DMA_DCHPRI58)	8	R/W	See section	16.4.26/ 711
13B	Channel n Priority Register (DMA_DCHPRI59)	8	R/W	See section	16.4.26/ 711
13C	Channel n Priority Register (DMA_DCHPRI60)	8	R/W	See section	16.4.26/ 711
13D	Channel n Priority Register (DMA_DCHPRI61)	8	R/W	See section	16.4.26/ 711
13E	Channel n Priority Register (DMA_DCHPRI62)	8	R/W	See section	16.4.26/ 711
13F	Channel n Priority Register (DMA_DCHPRI63)	8	R/W	See section	16.4.26/ 711
1000	TCD Source Address (DMA_TCD0_SADDR)	32	R/W	0000_0000h	16.4.27/ 712
1004	TCD Transfer Attributes (DMA_TCD0_ATTR)	16	R/W	0000h	16.4.28/ 712
1006	TCD Signed Source Address Offset (DMA_TCD0_SOFF)	16	R/W	0000h	16.4.29/ 713

Table continues on the next page...

DMA memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/page
1008	TCD Minor Byte Count (Minor Loop Mapping Disabled) (DMA_TCD0_NBYTES_MLNO)	32	R/W	0000_0000h	16.4.30/713
1008	TCD Signed Minor Loop Offset (Minor Loop Mapping Enabled and Offset Disabled) (DMA_TCD0_NBYTES_MLOFFNO)	32	R/W	0000_0000h	16.4.31/714
1008	TCD Signed Minor Loop Offset (Minor Loop Mapping and Offset Enabled) (DMA_TCD0_NBYTES_MLOFFYES)	32	R/W	0000_0000h	16.4.32/715
100C	TCD Last Source Address Adjustment (DMA_TCD0_SLAST)	32	R/W	0000_0000h	16.4.33/716
1010	TCD Destination Address (DMA_TCD0_DADDR)	32	R/W	0000_0000h	16.4.34/717
1014	TCD Current Minor Loop Link, Major Loop Count (Channel Linking Enabled) (DMA_TCD0_CITER_ELINKYES)	16	R/W	0000h	16.4.35/717
1014	DMA_TCD0_CITER_ELINKNO	16	R/W	0000h	16.4.36/718
1016	TCD Signed Destination Address Offset (DMA_TCD0_DOFF)	16	R/W	0000h	16.4.37/719
1018	TCD Last Destination Address Adjustment/Scatter Gather Address (DMA_TCD0_DLASTGA)	32	R/W	0000_0000h	16.4.38/720
101C	TCD Beginning Minor Loop Link, Major Loop Count (Channel Linking Enabled) (DMA_TCD0_BITER_ELINKYES)	16	R/W	0000h	16.4.39/720
101C	TCD Beginning Minor Loop Link, Major Loop Count (Channel Linking Disabled) (DMA_TCD0_BITER_ELINKNO)	16	R/W	0000h	16.4.40/721
101E	TCD Control and Status (DMA_TCD0_CSR)	16	R/W	0000h	16.4.41/722
1020	TCD Source Address (DMA_TCD1_SADDR)	32	R/W	0000_0000h	16.4.27/712
1024	TCD Transfer Attributes (DMA_TCD1_ATTR)	16	R/W	0000h	16.4.28/712
1026	TCD Signed Source Address Offset (DMA_TCD1_SOFF)	16	R/W	0000h	16.4.29/713
1028	TCD Minor Byte Count (Minor Loop Mapping Disabled) (DMA_TCD1_NBYTES_MLNO)	32	R/W	0000_0000h	16.4.30/713
1028	TCD Signed Minor Loop Offset (Minor Loop Mapping Enabled and Offset Disabled) (DMA_TCD1_NBYTES_MLOFFNO)	32	R/W	0000_0000h	16.4.31/714
1028	TCD Signed Minor Loop Offset (Minor Loop Mapping and Offset Enabled) (DMA_TCD1_NBYTES_MLOFFYES)	32	R/W	0000_0000h	16.4.32/715
102C	TCD Last Source Address Adjustment (DMA_TCD1_SLAST)	32	R/W	0000_0000h	16.4.33/716
1030	TCD Destination Address (DMA_TCD1_DADDR)	32	R/W	0000_0000h	16.4.34/717
1034	TCD Current Minor Loop Link, Major Loop Count (Channel Linking Enabled) (DMA_TCD1_CITER_ELINKYES)	16	R/W	0000h	16.4.35/717

Table continues on the next page...

DMA memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
1034	DMA_TCD1_CITER_ELINKNO	16	R/W	0000h	16.4.36/ 718
1036	TCD Signed Destination Address Offset (DMA_TCD1_DOFF)	16	R/W	0000h	16.4.37/ 719
1038	TCD Last Destination Address Adjustment/Scatter Gather Address (DMA_TCD1_DLASTSGA)	32	R/W	0000_0000h	16.4.38/ 720
103C	TCD Beginning Minor Loop Link, Major Loop Count (Channel Linking Enabled) (DMA_TCD1_BITER_ELINKYES)	16	R/W	0000h	16.4.39/ 720
103C	TCD Beginning Minor Loop Link, Major Loop Count (Channel Linking Disabled) (DMA_TCD1_BITER_ELINKNO)	16	R/W	0000h	16.4.40/ 721
103E	TCD Control and Status (DMA_TCD1_CSR)	16	R/W	0000h	16.4.41/ 722
1040	TCD Source Address (DMA_TCD2_SADDR)	32	R/W	0000_0000h	16.4.27/ 712
1044	TCD Transfer Attributes (DMA_TCD2_ATTR)	16	R/W	0000h	16.4.28/ 712
1046	TCD Signed Source Address Offset (DMA_TCD2_SOFF)	16	R/W	0000h	16.4.29/ 713
1048	TCD Minor Byte Count (Minor Loop Mapping Disabled) (DMA_TCD2_NBYTES_MLNO)	32	R/W	0000_0000h	16.4.30/ 713
1048	TCD Signed Minor Loop Offset (Minor Loop Mapping Enabled and Offset Disabled) (DMA_TCD2_NBYTES_MLOFFNO)	32	R/W	0000_0000h	16.4.31/ 714
1048	TCD Signed Minor Loop Offset (Minor Loop Mapping and Offset Enabled) (DMA_TCD2_NBYTES_MLOFFYES)	32	R/W	0000_0000h	16.4.32/ 715
104C	TCD Last Source Address Adjustment (DMA_TCD2_SLAST)	32	R/W	0000_0000h	16.4.33/ 716
1050	TCD Destination Address (DMA_TCD2_DADDR)	32	R/W	0000_0000h	16.4.34/ 717
1054	TCD Current Minor Loop Link, Major Loop Count (Channel Linking Enabled) (DMA_TCD2_CITER_ELINKYES)	16	R/W	0000h	16.4.35/ 717
1054	DMA_TCD2_CITER_ELINKNO	16	R/W	0000h	16.4.36/ 718
1056	TCD Signed Destination Address Offset (DMA_TCD2_DOFF)	16	R/W	0000h	16.4.37/ 719
1058	TCD Last Destination Address Adjustment/Scatter Gather Address (DMA_TCD2_DLASTSGA)	32	R/W	0000_0000h	16.4.38/ 720
105C	TCD Beginning Minor Loop Link, Major Loop Count (Channel Linking Enabled) (DMA_TCD2_BITER_ELINKYES)	16	R/W	0000h	16.4.39/ 720
105C	TCD Beginning Minor Loop Link, Major Loop Count (Channel Linking Disabled) (DMA_TCD2_BITER_ELINKNO)	16	R/W	0000h	16.4.40/ 721
105E	TCD Control and Status (DMA_TCD2_CSR)	16	R/W	0000h	16.4.41/ 722

Table continues on the next page...

DMA memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
1060	TCD Source Address (DMA_TCD3_SADDR)	32	R/W	0000_0000h	16.4.27/ 712
1064	TCD Transfer Attributes (DMA_TCD3_ATTR)	16	R/W	0000h	16.4.28/ 712
1066	TCD Signed Source Address Offset (DMA_TCD3_SOFF)	16	R/W	0000h	16.4.29/ 713
1068	TCD Minor Byte Count (Minor Loop Mapping Disabled) (DMA_TCD3_NBYTES_MLNO)	32	R/W	0000_0000h	16.4.30/ 713
1068	TCD Signed Minor Loop Offset (Minor Loop Mapping Enabled and Offset Disabled) (DMA_TCD3_NBYTES_MLOFFNO)	32	R/W	0000_0000h	16.4.31/ 714
1068	TCD Signed Minor Loop Offset (Minor Loop Mapping and Offset Enabled) (DMA_TCD3_NBYTES_MLOFFYES)	32	R/W	0000_0000h	16.4.32/ 715
106C	TCD Last Source Address Adjustment (DMA_TCD3_SLAST)	32	R/W	0000_0000h	16.4.33/ 716
1070	TCD Destination Address (DMA_TCD3_DADDR)	32	R/W	0000_0000h	16.4.34/ 717
1074	TCD Current Minor Loop Link, Major Loop Count (Channel Linking Enabled) (DMA_TCD3_CITER_ELINKYES)	16	R/W	0000h	16.4.35/ 717
1074	DMA_TCD3_CITER_ELINKNO	16	R/W	0000h	16.4.36/ 718
1076	TCD Signed Destination Address Offset (DMA_TCD3_DOFF)	16	R/W	0000h	16.4.37/ 719
1078	TCD Last Destination Address Adjustment/Scatter Gather Address (DMA_TCD3_DLASTGA)	32	R/W	0000_0000h	16.4.38/ 720
107C	TCD Beginning Minor Loop Link, Major Loop Count (Channel Linking Enabled) (DMA_TCD3_BITER_ELINKYES)	16	R/W	0000h	16.4.39/ 720
107C	TCD Beginning Minor Loop Link, Major Loop Count (Channel Linking Disabled) (DMA_TCD3_BITER_ELINKNO)	16	R/W	0000h	16.4.40/ 721
107E	TCD Control and Status (DMA_TCD3_CSR)	16	R/W	0000h	16.4.41/ 722
1080	TCD Source Address (DMA_TCD4_SADDR)	32	R/W	0000_0000h	16.4.27/ 712
1084	TCD Transfer Attributes (DMA_TCD4_ATTR)	16	R/W	0000h	16.4.28/ 712
1086	TCD Signed Source Address Offset (DMA_TCD4_SOFF)	16	R/W	0000h	16.4.29/ 713
1088	TCD Minor Byte Count (Minor Loop Mapping Disabled) (DMA_TCD4_NBYTES_MLNO)	32	R/W	0000_0000h	16.4.30/ 713
1088	TCD Signed Minor Loop Offset (Minor Loop Mapping Enabled and Offset Disabled) (DMA_TCD4_NBYTES_MLOFFNO)	32	R/W	0000_0000h	16.4.31/ 714
1088	TCD Signed Minor Loop Offset (Minor Loop Mapping and Offset Enabled) (DMA_TCD4_NBYTES_MLOFFYES)	32	R/W	0000_0000h	16.4.32/ 715

Table continues on the next page...

DMA memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
108C	TCD Last Source Address Adjustment (DMA_TCD4_SLAST)	32	R/W	0000_0000h	16.4.33/ 716
1090	TCD Destination Address (DMA_TCD4_DADDR)	32	R/W	0000_0000h	16.4.34/ 717
1094	TCD Current Minor Loop Link, Major Loop Count (Channel Linking Enabled) (DMA_TCD4_CITER_ELINKYES)	16	R/W	0000h	16.4.35/ 717
1094	DMA_TCD4_CITER_ELINKNO	16	R/W	0000h	16.4.36/ 718
1096	TCD Signed Destination Address Offset (DMA_TCD4_DOFF)	16	R/W	0000h	16.4.37/ 719
1098	TCD Last Destination Address Adjustment/Scatter Gather Address (DMA_TCD4_DLASTSGA)	32	R/W	0000_0000h	16.4.38/ 720
109C	TCD Beginning Minor Loop Link, Major Loop Count (Channel Linking Enabled) (DMA_TCD4_BITER_ELINKYES)	16	R/W	0000h	16.4.39/ 720
109C	TCD Beginning Minor Loop Link, Major Loop Count (Channel Linking Disabled) (DMA_TCD4_BITER_ELINKNO)	16	R/W	0000h	16.4.40/ 721
109E	TCD Control and Status (DMA_TCD4_CSR)	16	R/W	0000h	16.4.41/ 722
10A0	TCD Source Address (DMA_TCD5_SADDR)	32	R/W	0000_0000h	16.4.27/ 712
10A4	TCD Transfer Attributes (DMA_TCD5_ATTR)	16	R/W	0000h	16.4.28/ 712
10A6	TCD Signed Source Address Offset (DMA_TCD5_SOFF)	16	R/W	0000h	16.4.29/ 713
10A8	TCD Minor Byte Count (Minor Loop Mapping Disabled) (DMA_TCD5_NBYTES_MLNO)	32	R/W	0000_0000h	16.4.30/ 713
10A8	TCD Signed Minor Loop Offset (Minor Loop Mapping Enabled and Offset Disabled) (DMA_TCD5_NBYTES_MLOFFNO)	32	R/W	0000_0000h	16.4.31/ 714
10A8	TCD Signed Minor Loop Offset (Minor Loop Mapping and Offset Enabled) (DMA_TCD5_NBYTES_MLOFFYES)	32	R/W	0000_0000h	16.4.32/ 715
10AC	TCD Last Source Address Adjustment (DMA_TCD5_SLAST)	32	R/W	0000_0000h	16.4.33/ 716
10B0	TCD Destination Address (DMA_TCD5_DADDR)	32	R/W	0000_0000h	16.4.34/ 717
10B4	TCD Current Minor Loop Link, Major Loop Count (Channel Linking Enabled) (DMA_TCD5_CITER_ELINKYES)	16	R/W	0000h	16.4.35/ 717
10B4	DMA_TCD5_CITER_ELINKNO	16	R/W	0000h	16.4.36/ 718
10B6	TCD Signed Destination Address Offset (DMA_TCD5_DOFF)	16	R/W	0000h	16.4.37/ 719
10B8	TCD Last Destination Address Adjustment/Scatter Gather Address (DMA_TCD5_DLASTSGA)	32	R/W	0000_0000h	16.4.38/ 720

Table continues on the next page...

DMA memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
10BC	TCD Beginning Minor Loop Link, Major Loop Count (Channel Linking Enabled) (DMA_TCD5_BITER_ELINKYES)	16	R/W	0000h	16.4.39/ 720
10BC	TCD Beginning Minor Loop Link, Major Loop Count (Channel Linking Disabled) (DMA_TCD5_BITER_ELINKNO)	16	R/W	0000h	16.4.40/ 721
10BE	TCD Control and Status (DMA_TCD5_CSR)	16	R/W	0000h	16.4.41/ 722
10C0	TCD Source Address (DMA_TCD6_SADDR)	32	R/W	0000_0000h	16.4.27/ 712
10C4	TCD Transfer Attributes (DMA_TCD6_ATTR)	16	R/W	0000h	16.4.28/ 712
10C6	TCD Signed Source Address Offset (DMA_TCD6_SOFF)	16	R/W	0000h	16.4.29/ 713
10C8	TCD Minor Byte Count (Minor Loop Mapping Disabled) (DMA_TCD6_NBYTES_MLNO)	32	R/W	0000_0000h	16.4.30/ 713
10C8	TCD Signed Minor Loop Offset (Minor Loop Mapping Enabled and Offset Disabled) (DMA_TCD6_NBYTES_MLOFFNO)	32	R/W	0000_0000h	16.4.31/ 714
10C8	TCD Signed Minor Loop Offset (Minor Loop Mapping and Offset Enabled) (DMA_TCD6_NBYTES_MLOFFYES)	32	R/W	0000_0000h	16.4.32/ 715
10CC	TCD Last Source Address Adjustment (DMA_TCD6_SLAST)	32	R/W	0000_0000h	16.4.33/ 716
10D0	TCD Destination Address (DMA_TCD6_DADDR)	32	R/W	0000_0000h	16.4.34/ 717
10D4	TCD Current Minor Loop Link, Major Loop Count (Channel Linking Enabled) (DMA_TCD6_CITER_ELINKYES)	16	R/W	0000h	16.4.35/ 717
10D4	DMA_TCD6_CITER_ELINKNO	16	R/W	0000h	16.4.36/ 718
10D6	TCD Signed Destination Address Offset (DMA_TCD6_DOFF)	16	R/W	0000h	16.4.37/ 719
10D8	TCD Last Destination Address Adjustment/Scatter Gather Address (DMA_TCD6_DLASTSGA)	32	R/W	0000_0000h	16.4.38/ 720
10DC	TCD Beginning Minor Loop Link, Major Loop Count (Channel Linking Enabled) (DMA_TCD6_BITER_ELINKYES)	16	R/W	0000h	16.4.39/ 720
10DC	TCD Beginning Minor Loop Link, Major Loop Count (Channel Linking Disabled) (DMA_TCD6_BITER_ELINKNO)	16	R/W	0000h	16.4.40/ 721
10DE	TCD Control and Status (DMA_TCD6_CSR)	16	R/W	0000h	16.4.41/ 722
10E0	TCD Source Address (DMA_TCD7_SADDR)	32	R/W	0000_0000h	16.4.27/ 712
10E4	TCD Transfer Attributes (DMA_TCD7_ATTR)	16	R/W	0000h	16.4.28/ 712
10E6	TCD Signed Source Address Offset (DMA_TCD7_SOFF)	16	R/W	0000h	16.4.29/ 713

Table continues on the next page...

DMA memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
10E8	TCD Minor Byte Count (Minor Loop Mapping Disabled) (DMA_TCD7_NBYTES_MLNO)	32	R/W	0000_0000h	16.4.30/ 713
10E8	TCD Signed Minor Loop Offset (Minor Loop Mapping Enabled and Offset Disabled) (DMA_TCD7_NBYTES_MLOFFNO)	32	R/W	0000_0000h	16.4.31/ 714
10E8	TCD Signed Minor Loop Offset (Minor Loop Mapping and Offset Enabled) (DMA_TCD7_NBYTES_MLOFFYES)	32	R/W	0000_0000h	16.4.32/ 715
10EC	TCD Last Source Address Adjustment (DMA_TCD7_SLAST)	32	R/W	0000_0000h	16.4.33/ 716
10F0	TCD Destination Address (DMA_TCD7_DADDR)	32	R/W	0000_0000h	16.4.34/ 717
10F4	TCD Current Minor Loop Link, Major Loop Count (Channel Linking Enabled) (DMA_TCD7_CITER_ELINKYES)	16	R/W	0000h	16.4.35/ 717
10F4	DMA_TCD7_CITER_ELINKNO	16	R/W	0000h	16.4.36/ 718
10F6	TCD Signed Destination Address Offset (DMA_TCD7_DOFF)	16	R/W	0000h	16.4.37/ 719
10F8	TCD Last Destination Address Adjustment/Scatter Gather Address (DMA_TCD7_DLASTSGA)	32	R/W	0000_0000h	16.4.38/ 720
10FC	TCD Beginning Minor Loop Link, Major Loop Count (Channel Linking Enabled) (DMA_TCD7_BITER_ELINKYES)	16	R/W	0000h	16.4.39/ 720
10FC	TCD Beginning Minor Loop Link, Major Loop Count (Channel Linking Disabled) (DMA_TCD7_BITER_ELINKNO)	16	R/W	0000h	16.4.40/ 721
10FE	TCD Control and Status (DMA_TCD7_CSR)	16	R/W	0000h	16.4.41/ 722
1100	TCD Source Address (DMA_TCD8_SADDR)	32	R/W	0000_0000h	16.4.27/ 712
1104	TCD Transfer Attributes (DMA_TCD8_ATTR)	16	R/W	0000h	16.4.28/ 712
1106	TCD Signed Source Address Offset (DMA_TCD8_SOFF)	16	R/W	0000h	16.4.29/ 713
1108	TCD Minor Byte Count (Minor Loop Mapping Disabled) (DMA_TCD8_NBYTES_MLNO)	32	R/W	0000_0000h	16.4.30/ 713
1108	TCD Signed Minor Loop Offset (Minor Loop Mapping Enabled and Offset Disabled) (DMA_TCD8_NBYTES_MLOFFNO)	32	R/W	0000_0000h	16.4.31/ 714
1108	TCD Signed Minor Loop Offset (Minor Loop Mapping and Offset Enabled) (DMA_TCD8_NBYTES_MLOFFYES)	32	R/W	0000_0000h	16.4.32/ 715
110C	TCD Last Source Address Adjustment (DMA_TCD8_SLAST)	32	R/W	0000_0000h	16.4.33/ 716
1110	TCD Destination Address (DMA_TCD8_DADDR)	32	R/W	0000_0000h	16.4.34/ 717
1114	TCD Current Minor Loop Link, Major Loop Count (Channel Linking Enabled) (DMA_TCD8_CITER_ELINKYES)	16	R/W	0000h	16.4.35/ 717

Table continues on the next page...

DMA memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
1114	DMA_TCD8_CITER_ELINKNO	16	R/W	0000h	16.4.36/ 718
1116	TCD Signed Destination Address Offset (DMA_TCD8_DOFF)	16	R/W	0000h	16.4.37/ 719
1118	TCD Last Destination Address Adjustment/Scatter Gather Address (DMA_TCD8_DLASTSGA)	32	R/W	0000_0000h	16.4.38/ 720
111C	TCD Beginning Minor Loop Link, Major Loop Count (Channel Linking Enabled) (DMA_TCD8_BITER_ELINKYES)	16	R/W	0000h	16.4.39/ 720
111C	TCD Beginning Minor Loop Link, Major Loop Count (Channel Linking Disabled) (DMA_TCD8_BITER_ELINKNO)	16	R/W	0000h	16.4.40/ 721
111E	TCD Control and Status (DMA_TCD8_CSR)	16	R/W	0000h	16.4.41/ 722
1120	TCD Source Address (DMA_TCD9_SADDR)	32	R/W	0000_0000h	16.4.27/ 712
1124	TCD Transfer Attributes (DMA_TCD9_ATTR)	16	R/W	0000h	16.4.28/ 712
1126	TCD Signed Source Address Offset (DMA_TCD9_SOFF)	16	R/W	0000h	16.4.29/ 713
1128	TCD Minor Byte Count (Minor Loop Mapping Disabled) (DMA_TCD9_NBYTES_MLNO)	32	R/W	0000_0000h	16.4.30/ 713
1128	TCD Signed Minor Loop Offset (Minor Loop Mapping Enabled and Offset Disabled) (DMA_TCD9_NBYTES_MLOFFNO)	32	R/W	0000_0000h	16.4.31/ 714
1128	TCD Signed Minor Loop Offset (Minor Loop Mapping and Offset Enabled) (DMA_TCD9_NBYTES_MLOFFYES)	32	R/W	0000_0000h	16.4.32/ 715
112C	TCD Last Source Address Adjustment (DMA_TCD9_SLAST)	32	R/W	0000_0000h	16.4.33/ 716
1130	TCD Destination Address (DMA_TCD9_DADDR)	32	R/W	0000_0000h	16.4.34/ 717
1134	TCD Current Minor Loop Link, Major Loop Count (Channel Linking Enabled) (DMA_TCD9_CITER_ELINKYES)	16	R/W	0000h	16.4.35/ 717
1134	DMA_TCD9_CITER_ELINKNO	16	R/W	0000h	16.4.36/ 718
1136	TCD Signed Destination Address Offset (DMA_TCD9_DOFF)	16	R/W	0000h	16.4.37/ 719
1138	TCD Last Destination Address Adjustment/Scatter Gather Address (DMA_TCD9_DLASTSGA)	32	R/W	0000_0000h	16.4.38/ 720
113C	TCD Beginning Minor Loop Link, Major Loop Count (Channel Linking Enabled) (DMA_TCD9_BITER_ELINKYES)	16	R/W	0000h	16.4.39/ 720
113C	TCD Beginning Minor Loop Link, Major Loop Count (Channel Linking Disabled) (DMA_TCD9_BITER_ELINKNO)	16	R/W	0000h	16.4.40/ 721
113E	TCD Control and Status (DMA_TCD9_CSR)	16	R/W	0000h	16.4.41/ 722

Table continues on the next page...

DMA memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
1140	TCD Source Address (DMA_TCD10_SADDR)	32	R/W	0000_0000h	16.4.27/ 712
1144	TCD Transfer Attributes (DMA_TCD10_ATTR)	16	R/W	0000h	16.4.28/ 712
1146	TCD Signed Source Address Offset (DMA_TCD10_SOFF)	16	R/W	0000h	16.4.29/ 713
1148	TCD Minor Byte Count (Minor Loop Mapping Disabled) (DMA_TCD10_NBYTES_MLNO)	32	R/W	0000_0000h	16.4.30/ 713
1148	TCD Signed Minor Loop Offset (Minor Loop Mapping Enabled and Offset Disabled) (DMA_TCD10_NBYTES_MLOFFNO)	32	R/W	0000_0000h	16.4.31/ 714
1148	TCD Signed Minor Loop Offset (Minor Loop Mapping and Offset Enabled) (DMA_TCD10_NBYTES_MLOFFYES)	32	R/W	0000_0000h	16.4.32/ 715
114C	TCD Last Source Address Adjustment (DMA_TCD10_SLAST)	32	R/W	0000_0000h	16.4.33/ 716
1150	TCD Destination Address (DMA_TCD10_DADDR)	32	R/W	0000_0000h	16.4.34/ 717
1154	TCD Current Minor Loop Link, Major Loop Count (Channel Linking Enabled) (DMA_TCD10_CITER_ELINKYES)	16	R/W	0000h	16.4.35/ 717
1154	DMA_TCD10_CITER_ELINKNO	16	R/W	0000h	16.4.36/ 718
1156	TCD Signed Destination Address Offset (DMA_TCD10_DOFF)	16	R/W	0000h	16.4.37/ 719
1158	TCD Last Destination Address Adjustment/Scatter Gather Address (DMA_TCD10_DLASTSGA)	32	R/W	0000_0000h	16.4.38/ 720
115C	TCD Beginning Minor Loop Link, Major Loop Count (Channel Linking Enabled) (DMA_TCD10_BITER_ELINKYES)	16	R/W	0000h	16.4.39/ 720
115C	TCD Beginning Minor Loop Link, Major Loop Count (Channel Linking Disabled) (DMA_TCD10_BITER_ELINKNO)	16	R/W	0000h	16.4.40/ 721
115E	TCD Control and Status (DMA_TCD10_CSR)	16	R/W	0000h	16.4.41/ 722
1160	TCD Source Address (DMA_TCD11_SADDR)	32	R/W	0000_0000h	16.4.27/ 712
1164	TCD Transfer Attributes (DMA_TCD11_ATTR)	16	R/W	0000h	16.4.28/ 712
1166	TCD Signed Source Address Offset (DMA_TCD11_SOFF)	16	R/W	0000h	16.4.29/ 713
1168	TCD Minor Byte Count (Minor Loop Mapping Disabled) (DMA_TCD11_NBYTES_MLNO)	32	R/W	0000_0000h	16.4.30/ 713
1168	TCD Signed Minor Loop Offset (Minor Loop Mapping Enabled and Offset Disabled) (DMA_TCD11_NBYTES_MLOFFNO)	32	R/W	0000_0000h	16.4.31/ 714
1168	TCD Signed Minor Loop Offset (Minor Loop Mapping and Offset Enabled) (DMA_TCD11_NBYTES_MLOFFYES)	32	R/W	0000_0000h	16.4.32/ 715

Table continues on the next page...

DMA memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
116C	TCD Last Source Address Adjustment (DMA_TCD11_SLAST)	32	R/W	0000_0000h	16.4.33/ 716
1170	TCD Destination Address (DMA_TCD11_DADDR)	32	R/W	0000_0000h	16.4.34/ 717
1174	TCD Current Minor Loop Link, Major Loop Count (Channel Linking Enabled) (DMA_TCD11_CITER_ELINKYES)	16	R/W	0000h	16.4.35/ 717
1174	DMA_TCD11_CITER_ELINKNO	16	R/W	0000h	16.4.36/ 718
1176	TCD Signed Destination Address Offset (DMA_TCD11_DOFF)	16	R/W	0000h	16.4.37/ 719
1178	TCD Last Destination Address Adjustment/Scatter Gather Address (DMA_TCD11_DLASTSGA)	32	R/W	0000_0000h	16.4.38/ 720
117C	TCD Beginning Minor Loop Link, Major Loop Count (Channel Linking Enabled) (DMA_TCD11_BITER_ELINKYES)	16	R/W	0000h	16.4.39/ 720
117C	TCD Beginning Minor Loop Link, Major Loop Count (Channel Linking Disabled) (DMA_TCD11_BITER_ELINKNO)	16	R/W	0000h	16.4.40/ 721
117E	TCD Control and Status (DMA_TCD11_CSR)	16	R/W	0000h	16.4.41/ 722
1180	TCD Source Address (DMA_TCD12_SADDR)	32	R/W	0000_0000h	16.4.27/ 712
1184	TCD Transfer Attributes (DMA_TCD12_ATTR)	16	R/W	0000h	16.4.28/ 712
1186	TCD Signed Source Address Offset (DMA_TCD12_SOFF)	16	R/W	0000h	16.4.29/ 713
1188	TCD Minor Byte Count (Minor Loop Mapping Disabled) (DMA_TCD12_NBYTES_MLNO)	32	R/W	0000_0000h	16.4.30/ 713
1188	TCD Signed Minor Loop Offset (Minor Loop Mapping Enabled and Offset Disabled) (DMA_TCD12_NBYTES_MLOFFNO)	32	R/W	0000_0000h	16.4.31/ 714
1188	TCD Signed Minor Loop Offset (Minor Loop Mapping and Offset Enabled) (DMA_TCD12_NBYTES_MLOFFYES)	32	R/W	0000_0000h	16.4.32/ 715
118C	TCD Last Source Address Adjustment (DMA_TCD12_SLAST)	32	R/W	0000_0000h	16.4.33/ 716
1190	TCD Destination Address (DMA_TCD12_DADDR)	32	R/W	0000_0000h	16.4.34/ 717
1194	TCD Current Minor Loop Link, Major Loop Count (Channel Linking Enabled) (DMA_TCD12_CITER_ELINKYES)	16	R/W	0000h	16.4.35/ 717
1194	DMA_TCD12_CITER_ELINKNO	16	R/W	0000h	16.4.36/ 718
1196	TCD Signed Destination Address Offset (DMA_TCD12_DOFF)	16	R/W	0000h	16.4.37/ 719
1198	TCD Last Destination Address Adjustment/Scatter Gather Address (DMA_TCD12_DLASTSGA)	32	R/W	0000_0000h	16.4.38/ 720

Table continues on the next page...

DMA memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
119C	TCD Beginning Minor Loop Link, Major Loop Count (Channel Linking Enabled) (DMA_TCD12_BITER_ELINKYES)	16	R/W	0000h	16.4.39/ 720
119C	TCD Beginning Minor Loop Link, Major Loop Count (Channel Linking Disabled) (DMA_TCD12_BITER_ELINKNO)	16	R/W	0000h	16.4.40/ 721
119E	TCD Control and Status (DMA_TCD12_CSR)	16	R/W	0000h	16.4.41/ 722
11A0	TCD Source Address (DMA_TCD13_SADDR)	32	R/W	0000_0000h	16.4.27/ 712
11A4	TCD Transfer Attributes (DMA_TCD13_ATTR)	16	R/W	0000h	16.4.28/ 712
11A6	TCD Signed Source Address Offset (DMA_TCD13_SOFF)	16	R/W	0000h	16.4.29/ 713
11A8	TCD Minor Byte Count (Minor Loop Mapping Disabled) (DMA_TCD13_NBYTES_MLNO)	32	R/W	0000_0000h	16.4.30/ 713
11A8	TCD Signed Minor Loop Offset (Minor Loop Mapping Enabled and Offset Disabled) (DMA_TCD13_NBYTES_MLOFFNO)	32	R/W	0000_0000h	16.4.31/ 714
11A8	TCD Signed Minor Loop Offset (Minor Loop Mapping and Offset Enabled) (DMA_TCD13_NBYTES_MLOFFYES)	32	R/W	0000_0000h	16.4.32/ 715
11AC	TCD Last Source Address Adjustment (DMA_TCD13_SLAST)	32	R/W	0000_0000h	16.4.33/ 716
11B0	TCD Destination Address (DMA_TCD13_DADDR)	32	R/W	0000_0000h	16.4.34/ 717
11B4	TCD Current Minor Loop Link, Major Loop Count (Channel Linking Enabled) (DMA_TCD13_CITER_ELINKYES)	16	R/W	0000h	16.4.35/ 717
11B4	DMA_TCD13_CITER_ELINKNO	16	R/W	0000h	16.4.36/ 718
11B6	TCD Signed Destination Address Offset (DMA_TCD13_DOFF)	16	R/W	0000h	16.4.37/ 719
11B8	TCD Last Destination Address Adjustment/Scatter Gather Address (DMA_TCD13_DLASTSGA)	32	R/W	0000_0000h	16.4.38/ 720
11BC	TCD Beginning Minor Loop Link, Major Loop Count (Channel Linking Enabled) (DMA_TCD13_BITER_ELINKYES)	16	R/W	0000h	16.4.39/ 720
11BC	TCD Beginning Minor Loop Link, Major Loop Count (Channel Linking Disabled) (DMA_TCD13_BITER_ELINKNO)	16	R/W	0000h	16.4.40/ 721
11BE	TCD Control and Status (DMA_TCD13_CSR)	16	R/W	0000h	16.4.41/ 722
11C0	TCD Source Address (DMA_TCD14_SADDR)	32	R/W	0000_0000h	16.4.27/ 712
11C4	TCD Transfer Attributes (DMA_TCD14_ATTR)	16	R/W	0000h	16.4.28/ 712

Table continues on the next page...

DMA memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
11C6	TCD Signed Source Address Offset (DMA_TCD14_SOFF)	16	R/W	0000h	16.4.29/ 713
11C8	TCD Minor Byte Count (Minor Loop Mapping Disabled) (DMA_TCD14_NBYTES_MLNO)	32	R/W	0000_0000h	16.4.30/ 713
11C8	TCD Signed Minor Loop Offset (Minor Loop Mapping Enabled and Offset Disabled) (DMA_TCD14_NBYTES_MLOFFNO)	32	R/W	0000_0000h	16.4.31/ 714
11C8	TCD Signed Minor Loop Offset (Minor Loop Mapping and Offset Enabled) (DMA_TCD14_NBYTES_MLOFFYES)	32	R/W	0000_0000h	16.4.32/ 715
11CC	TCD Last Source Address Adjustment (DMA_TCD14_SLAST)	32	R/W	0000_0000h	16.4.33/ 716
11D0	TCD Destination Address (DMA_TCD14_DADDR)	32	R/W	0000_0000h	16.4.34/ 717
11D4	TCD Current Minor Loop Link, Major Loop Count (Channel Linking Enabled) (DMA_TCD14_CITER_ELINKYES)	16	R/W	0000h	16.4.35/ 717
11D4	DMA_TCD14_CITER_ELINKNO	16	R/W	0000h	16.4.36/ 718
11D6	TCD Signed Destination Address Offset (DMA_TCD14_DOFF)	16	R/W	0000h	16.4.37/ 719
11D8	TCD Last Destination Address Adjustment/Scatter Gather Address (DMA_TCD14_DLASTSGA)	32	R/W	0000_0000h	16.4.38/ 720
11DC	TCD Beginning Minor Loop Link, Major Loop Count (Channel Linking Enabled) (DMA_TCD14_BITER_ELINKYES)	16	R/W	0000h	16.4.39/ 720
11DC	TCD Beginning Minor Loop Link, Major Loop Count (Channel Linking Disabled) (DMA_TCD14_BITER_ELINKNO)	16	R/W	0000h	16.4.40/ 721
11DE	TCD Control and Status (DMA_TCD14_CSR)	16	R/W	0000h	16.4.41/ 722
11E0	TCD Source Address (DMA_TCD15_SADDR)	32	R/W	0000_0000h	16.4.27/ 712
11E4	TCD Transfer Attributes (DMA_TCD15_ATTR)	16	R/W	0000h	16.4.28/ 712
11E6	TCD Signed Source Address Offset (DMA_TCD15_SOFF)	16	R/W	0000h	16.4.29/ 713
11E8	TCD Minor Byte Count (Minor Loop Mapping Disabled) (DMA_TCD15_NBYTES_MLNO)	32	R/W	0000_0000h	16.4.30/ 713
11E8	TCD Signed Minor Loop Offset (Minor Loop Mapping Enabled and Offset Disabled) (DMA_TCD15_NBYTES_MLOFFNO)	32	R/W	0000_0000h	16.4.31/ 714
11E8	TCD Signed Minor Loop Offset (Minor Loop Mapping and Offset Enabled) (DMA_TCD15_NBYTES_MLOFFYES)	32	R/W	0000_0000h	16.4.32/ 715
11EC	TCD Last Source Address Adjustment (DMA_TCD15_SLAST)	32	R/W	0000_0000h	16.4.33/ 716
11F0	TCD Destination Address (DMA_TCD15_DADDR)	32	R/W	0000_0000h	16.4.34/ 717

Table continues on the next page...

DMA memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
11F4	TCD Current Minor Loop Link, Major Loop Count (Channel Linking Enabled) (DMA_TCD15_CITER_ELINKYES)	16	R/W	0000h	16.4.35/ 717
11F4	DMA_TCD15_CITER_ELINKNO	16	R/W	0000h	16.4.36/ 718
11F6	TCD Signed Destination Address Offset (DMA_TCD15_DOFF)	16	R/W	0000h	16.4.37/ 719
11F8	TCD Last Destination Address Adjustment/Scatter Gather Address (DMA_TCD15_DLASTSGA)	32	R/W	0000_0000h	16.4.38/ 720
11FC	TCD Beginning Minor Loop Link, Major Loop Count (Channel Linking Enabled) (DMA_TCD15_BITER_ELINKYES)	16	R/W	0000h	16.4.39/ 720
11FC	TCD Beginning Minor Loop Link, Major Loop Count (Channel Linking Disabled) (DMA_TCD15_BITER_ELINKNO)	16	R/W	0000h	16.4.40/ 721
11FE	TCD Control and Status (DMA_TCD15_CSR)	16	R/W	0000h	16.4.41/ 722
1200	TCD Source Address (DMA_TCD16_SADDR)	32	R/W	0000_0000h	16.4.27/ 712
1204	TCD Transfer Attributes (DMA_TCD16_ATTR)	16	R/W	0000h	16.4.28/ 712
1206	TCD Signed Source Address Offset (DMA_TCD16_SOFF)	16	R/W	0000h	16.4.29/ 713
1208	TCD Minor Byte Count (Minor Loop Mapping Disabled) (DMA_TCD16_NBYTES_MLNO)	32	R/W	0000_0000h	16.4.30/ 713
1208	TCD Signed Minor Loop Offset (Minor Loop Mapping Enabled and Offset Disabled) (DMA_TCD16_NBYTES_MLOFFNO)	32	R/W	0000_0000h	16.4.31/ 714
1208	TCD Signed Minor Loop Offset (Minor Loop Mapping and Offset Enabled) (DMA_TCD16_NBYTES_MLOFFYES)	32	R/W	0000_0000h	16.4.32/ 715
120C	TCD Last Source Address Adjustment (DMA_TCD16_SLAST)	32	R/W	0000_0000h	16.4.33/ 716
1210	TCD Destination Address (DMA_TCD16_DADDR)	32	R/W	0000_0000h	16.4.34/ 717
1214	TCD Current Minor Loop Link, Major Loop Count (Channel Linking Enabled) (DMA_TCD16_CITER_ELINKYES)	16	R/W	0000h	16.4.35/ 717
1214	DMA_TCD16_CITER_ELINKNO	16	R/W	0000h	16.4.36/ 718
1216	TCD Signed Destination Address Offset (DMA_TCD16_DOFF)	16	R/W	0000h	16.4.37/ 719
1218	TCD Last Destination Address Adjustment/Scatter Gather Address (DMA_TCD16_DLASTSGA)	32	R/W	0000_0000h	16.4.38/ 720
121C	TCD Beginning Minor Loop Link, Major Loop Count (Channel Linking Enabled) (DMA_TCD16_BITER_ELINKYES)	16	R/W	0000h	16.4.39/ 720

Table continues on the next page...

DMA memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
121C	TCD Beginning Minor Loop Link, Major Loop Count (Channel Linking Disabled) (DMA_TCD16_BITER_ELINKNO)	16	R/W	0000h	16.4.40/ 721
121E	TCD Control and Status (DMA_TCD16_CSR)	16	R/W	0000h	16.4.41/ 722
1220	TCD Source Address (DMA_TCD17_SADDR)	32	R/W	0000_0000h	16.4.27/ 712
1224	TCD Transfer Attributes (DMA_TCD17_ATTR)	16	R/W	0000h	16.4.28/ 712
1226	TCD Signed Source Address Offset (DMA_TCD17_SOFF)	16	R/W	0000h	16.4.29/ 713
1228	TCD Minor Byte Count (Minor Loop Mapping Disabled) (DMA_TCD17_NBYTES_MLNO)	32	R/W	0000_0000h	16.4.30/ 713
1228	TCD Signed Minor Loop Offset (Minor Loop Mapping Enabled and Offset Disabled) (DMA_TCD17_NBYTES_MLOFFNO)	32	R/W	0000_0000h	16.4.31/ 714
1228	TCD Signed Minor Loop Offset (Minor Loop Mapping and Offset Enabled) (DMA_TCD17_NBYTES_MLOFFYES)	32	R/W	0000_0000h	16.4.32/ 715
122C	TCD Last Source Address Adjustment (DMA_TCD17_SLAST)	32	R/W	0000_0000h	16.4.33/ 716
1230	TCD Destination Address (DMA_TCD17_DADDR)	32	R/W	0000_0000h	16.4.34/ 717
1234	TCD Current Minor Loop Link, Major Loop Count (Channel Linking Enabled) (DMA_TCD17_CITER_ELINKYES)	16	R/W	0000h	16.4.35/ 717
1234	DMA_TCD17_CITER_ELINKNO	16	R/W	0000h	16.4.36/ 718
1236	TCD Signed Destination Address Offset (DMA_TCD17_DOFF)	16	R/W	0000h	16.4.37/ 719
1238	TCD Last Destination Address Adjustment/Scatter Gather Address (DMA_TCD17_DLASTSGA)	32	R/W	0000_0000h	16.4.38/ 720
123C	TCD Beginning Minor Loop Link, Major Loop Count (Channel Linking Enabled) (DMA_TCD17_BITER_ELINKYES)	16	R/W	0000h	16.4.39/ 720
123C	TCD Beginning Minor Loop Link, Major Loop Count (Channel Linking Disabled) (DMA_TCD17_BITER_ELINKNO)	16	R/W	0000h	16.4.40/ 721
123E	TCD Control and Status (DMA_TCD17_CSR)	16	R/W	0000h	16.4.41/ 722
1240	TCD Source Address (DMA_TCD18_SADDR)	32	R/W	0000_0000h	16.4.27/ 712
1244	TCD Transfer Attributes (DMA_TCD18_ATTR)	16	R/W	0000h	16.4.28/ 712
1246	TCD Signed Source Address Offset (DMA_TCD18_SOFF)	16	R/W	0000h	16.4.29/ 713
1248	TCD Minor Byte Count (Minor Loop Mapping Disabled) (DMA_TCD18_NBYTES_MLNO)	32	R/W	0000_0000h	16.4.30/ 713

Table continues on the next page...

DMA memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
1248	TCD Signed Minor Loop Offset (Minor Loop Mapping Enabled and Offset Disabled) (DMA_TCD18_NBYTES_MLOFFNO)	32	R/W	0000_0000h	16.4.31/ 714
1248	TCD Signed Minor Loop Offset (Minor Loop Mapping and Offset Enabled) (DMA_TCD18_NBYTES_MLOFFYES)	32	R/W	0000_0000h	16.4.32/ 715
124C	TCD Last Source Address Adjustment (DMA_TCD18_SLAST)	32	R/W	0000_0000h	16.4.33/ 716
1250	TCD Destination Address (DMA_TCD18_DADDR)	32	R/W	0000_0000h	16.4.34/ 717
1254	TCD Current Minor Loop Link, Major Loop Count (Channel Linking Enabled) (DMA_TCD18_CITER_ELINKYES)	16	R/W	0000h	16.4.35/ 717
1254	DMA_TCD18_CITER_ELINKNO	16	R/W	0000h	16.4.36/ 718
1256	TCD Signed Destination Address Offset (DMA_TCD18_DOFF)	16	R/W	0000h	16.4.37/ 719
1258	TCD Last Destination Address Adjustment/Scatter Gather Address (DMA_TCD18_DLASTSGA)	32	R/W	0000_0000h	16.4.38/ 720
125C	TCD Beginning Minor Loop Link, Major Loop Count (Channel Linking Enabled) (DMA_TCD18_BITER_ELINKYES)	16	R/W	0000h	16.4.39/ 720
125C	TCD Beginning Minor Loop Link, Major Loop Count (Channel Linking Disabled) (DMA_TCD18_BITER_ELINKNO)	16	R/W	0000h	16.4.40/ 721
125E	TCD Control and Status (DMA_TCD18_CSR)	16	R/W	0000h	16.4.41/ 722
1260	TCD Source Address (DMA_TCD19_SADDR)	32	R/W	0000_0000h	16.4.27/ 712
1264	TCD Transfer Attributes (DMA_TCD19_ATTR)	16	R/W	0000h	16.4.28/ 712
1266	TCD Signed Source Address Offset (DMA_TCD19_SOFF)	16	R/W	0000h	16.4.29/ 713
1268	TCD Minor Byte Count (Minor Loop Mapping Disabled) (DMA_TCD19_NBYTES_MLNO)	32	R/W	0000_0000h	16.4.30/ 713
1268	TCD Signed Minor Loop Offset (Minor Loop Mapping Enabled and Offset Disabled) (DMA_TCD19_NBYTES_MLOFFNO)	32	R/W	0000_0000h	16.4.31/ 714
1268	TCD Signed Minor Loop Offset (Minor Loop Mapping and Offset Enabled) (DMA_TCD19_NBYTES_MLOFFYES)	32	R/W	0000_0000h	16.4.32/ 715
126C	TCD Last Source Address Adjustment (DMA_TCD19_SLAST)	32	R/W	0000_0000h	16.4.33/ 716
1270	TCD Destination Address (DMA_TCD19_DADDR)	32	R/W	0000_0000h	16.4.34/ 717
1274	TCD Current Minor Loop Link, Major Loop Count (Channel Linking Enabled) (DMA_TCD19_CITER_ELINKYES)	16	R/W	0000h	16.4.35/ 717
1274	DMA_TCD19_CITER_ELINKNO	16	R/W	0000h	16.4.36/ 718

Table continues on the next page...

DMA memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
1276	TCD Signed Destination Address Offset (DMA_TCD19_DOFF)	16	R/W	0000h	16.4.37/ 719
1278	TCD Last Destination Address Adjustment/Scatter Gather Address (DMA_TCD19_DLASTSGA)	32	R/W	0000_0000h	16.4.38/ 720
127C	TCD Beginning Minor Loop Link, Major Loop Count (Channel Linking Enabled) (DMA_TCD19_BITER_ELINKYES)	16	R/W	0000h	16.4.39/ 720
127C	TCD Beginning Minor Loop Link, Major Loop Count (Channel Linking Disabled) (DMA_TCD19_BITER_ELINKNO)	16	R/W	0000h	16.4.40/ 721
127E	TCD Control and Status (DMA_TCD19_CSR)	16	R/W	0000h	16.4.41/ 722
1280	TCD Source Address (DMA_TCD20_SADDR)	32	R/W	0000_0000h	16.4.27/ 712
1284	TCD Transfer Attributes (DMA_TCD20_ATTR)	16	R/W	0000h	16.4.28/ 712
1286	TCD Signed Source Address Offset (DMA_TCD20_SOFF)	16	R/W	0000h	16.4.29/ 713
1288	TCD Minor Byte Count (Minor Loop Mapping Disabled) (DMA_TCD20_NBYTES_MLNO)	32	R/W	0000_0000h	16.4.30/ 713
1288	TCD Signed Minor Loop Offset (Minor Loop Mapping Enabled and Offset Disabled) (DMA_TCD20_NBYTES_MLOFFNO)	32	R/W	0000_0000h	16.4.31/ 714
1288	TCD Signed Minor Loop Offset (Minor Loop Mapping and Offset Enabled) (DMA_TCD20_NBYTES_MLOFFYES)	32	R/W	0000_0000h	16.4.32/ 715
128C	TCD Last Source Address Adjustment (DMA_TCD20_SLAST)	32	R/W	0000_0000h	16.4.33/ 716
1290	TCD Destination Address (DMA_TCD20_DADDR)	32	R/W	0000_0000h	16.4.34/ 717
1294	TCD Current Minor Loop Link, Major Loop Count (Channel Linking Enabled) (DMA_TCD20_CITER_ELINKYES)	16	R/W	0000h	16.4.35/ 717
1294	DMA_TCD20_CITER_ELINKNO	16	R/W	0000h	16.4.36/ 718
1296	TCD Signed Destination Address Offset (DMA_TCD20_DOFF)	16	R/W	0000h	16.4.37/ 719
1298	TCD Last Destination Address Adjustment/Scatter Gather Address (DMA_TCD20_DLASTSGA)	32	R/W	0000_0000h	16.4.38/ 720
129C	TCD Beginning Minor Loop Link, Major Loop Count (Channel Linking Enabled) (DMA_TCD20_BITER_ELINKYES)	16	R/W	0000h	16.4.39/ 720
129C	TCD Beginning Minor Loop Link, Major Loop Count (Channel Linking Disabled) (DMA_TCD20_BITER_ELINKNO)	16	R/W	0000h	16.4.40/ 721
129E	TCD Control and Status (DMA_TCD20_CSR)	16	R/W	0000h	16.4.41/ 722

Table continues on the next page...

DMA memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
12A0	TCD Source Address (DMA_TCD21_SADDR)	32	R/W	0000_0000h	16.4.27/ 712
12A4	TCD Transfer Attributes (DMA_TCD21_ATTR)	16	R/W	0000h	16.4.28/ 712
12A6	TCD Signed Source Address Offset (DMA_TCD21_SOFF)	16	R/W	0000h	16.4.29/ 713
12A8	TCD Minor Byte Count (Minor Loop Mapping Disabled) (DMA_TCD21_NBYTES_MLNO)	32	R/W	0000_0000h	16.4.30/ 713
12A8	TCD Signed Minor Loop Offset (Minor Loop Mapping Enabled and Offset Disabled) (DMA_TCD21_NBYTES_MLOFFNO)	32	R/W	0000_0000h	16.4.31/ 714
12A8	TCD Signed Minor Loop Offset (Minor Loop Mapping and Offset Enabled) (DMA_TCD21_NBYTES_MLOFFYES)	32	R/W	0000_0000h	16.4.32/ 715
12AC	TCD Last Source Address Adjustment (DMA_TCD21_SLAST)	32	R/W	0000_0000h	16.4.33/ 716
12B0	TCD Destination Address (DMA_TCD21_DADDR)	32	R/W	0000_0000h	16.4.34/ 717
12B4	TCD Current Minor Loop Link, Major Loop Count (Channel Linking Enabled) (DMA_TCD21_CITER_ELINKYES)	16	R/W	0000h	16.4.35/ 717
12B4	DMA_TCD21_CITER_ELINKNO	16	R/W	0000h	16.4.36/ 718
12B6	TCD Signed Destination Address Offset (DMA_TCD21_DOFF)	16	R/W	0000h	16.4.37/ 719
12B8	TCD Last Destination Address Adjustment/Scatter Gather Address (DMA_TCD21_DLASTSGA)	32	R/W	0000_0000h	16.4.38/ 720
12BC	TCD Beginning Minor Loop Link, Major Loop Count (Channel Linking Enabled) (DMA_TCD21_BITER_ELINKYES)	16	R/W	0000h	16.4.39/ 720
12BC	TCD Beginning Minor Loop Link, Major Loop Count (Channel Linking Disabled) (DMA_TCD21_BITER_ELINKNO)	16	R/W	0000h	16.4.40/ 721
12BE	TCD Control and Status (DMA_TCD21_CSR)	16	R/W	0000h	16.4.41/ 722
12C0	TCD Source Address (DMA_TCD22_SADDR)	32	R/W	0000_0000h	16.4.27/ 712
12C4	TCD Transfer Attributes (DMA_TCD22_ATTR)	16	R/W	0000h	16.4.28/ 712
12C6	TCD Signed Source Address Offset (DMA_TCD22_SOFF)	16	R/W	0000h	16.4.29/ 713
12C8	TCD Minor Byte Count (Minor Loop Mapping Disabled) (DMA_TCD22_NBYTES_MLNO)	32	R/W	0000_0000h	16.4.30/ 713
12C8	TCD Signed Minor Loop Offset (Minor Loop Mapping Enabled and Offset Disabled) (DMA_TCD22_NBYTES_MLOFFNO)	32	R/W	0000_0000h	16.4.31/ 714
12C8	TCD Signed Minor Loop Offset (Minor Loop Mapping and Offset Enabled) (DMA_TCD22_NBYTES_MLOFFYES)	32	R/W	0000_0000h	16.4.32/ 715

Table continues on the next page...

DMA memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
12CC	TCD Last Source Address Adjustment (DMA_TCD22_SLAST)	32	R/W	0000_0000h	16.4.33/ 716
12D0	TCD Destination Address (DMA_TCD22_DADDR)	32	R/W	0000_0000h	16.4.34/ 717
12D4	TCD Current Minor Loop Link, Major Loop Count (Channel Linking Enabled) (DMA_TCD22_CITER_ELINKYES)	16	R/W	0000h	16.4.35/ 717
12D4	DMA_TCD22_CITER_ELINKNO	16	R/W	0000h	16.4.36/ 718
12D6	TCD Signed Destination Address Offset (DMA_TCD22_DOFF)	16	R/W	0000h	16.4.37/ 719
12D8	TCD Last Destination Address Adjustment/Scatter Gather Address (DMA_TCD22_DLASTSGA)	32	R/W	0000_0000h	16.4.38/ 720
12DC	TCD Beginning Minor Loop Link, Major Loop Count (Channel Linking Enabled) (DMA_TCD22_BITER_ELINKYES)	16	R/W	0000h	16.4.39/ 720
12DC	TCD Beginning Minor Loop Link, Major Loop Count (Channel Linking Disabled) (DMA_TCD22_BITER_ELINKNO)	16	R/W	0000h	16.4.40/ 721
12DE	TCD Control and Status (DMA_TCD22_CSR)	16	R/W	0000h	16.4.41/ 722
12E0	TCD Source Address (DMA_TCD23_SADDR)	32	R/W	0000_0000h	16.4.27/ 712
12E4	TCD Transfer Attributes (DMA_TCD23_ATTR)	16	R/W	0000h	16.4.28/ 712
12E6	TCD Signed Source Address Offset (DMA_TCD23_SOFF)	16	R/W	0000h	16.4.29/ 713
12E8	TCD Minor Byte Count (Minor Loop Mapping Disabled) (DMA_TCD23_NBYTES_MLNO)	32	R/W	0000_0000h	16.4.30/ 713
12E8	TCD Signed Minor Loop Offset (Minor Loop Mapping Enabled and Offset Disabled) (DMA_TCD23_NBYTES_MLOFFNO)	32	R/W	0000_0000h	16.4.31/ 714
12E8	TCD Signed Minor Loop Offset (Minor Loop Mapping and Offset Enabled) (DMA_TCD23_NBYTES_MLOFFYES)	32	R/W	0000_0000h	16.4.32/ 715
12EC	TCD Last Source Address Adjustment (DMA_TCD23_SLAST)	32	R/W	0000_0000h	16.4.33/ 716
12F0	TCD Destination Address (DMA_TCD23_DADDR)	32	R/W	0000_0000h	16.4.34/ 717
12F4	TCD Current Minor Loop Link, Major Loop Count (Channel Linking Enabled) (DMA_TCD23_CITER_ELINKYES)	16	R/W	0000h	16.4.35/ 717
12F4	DMA_TCD23_CITER_ELINKNO	16	R/W	0000h	16.4.36/ 718
12F6	TCD Signed Destination Address Offset (DMA_TCD23_DOFF)	16	R/W	0000h	16.4.37/ 719
12F8	TCD Last Destination Address Adjustment/Scatter Gather Address (DMA_TCD23_DLASTSGA)	32	R/W	0000_0000h	16.4.38/ 720

Table continues on the next page...

DMA memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
12FC	TCD Beginning Minor Loop Link, Major Loop Count (Channel Linking Enabled) (DMA_TCD23_BITER_ELINKYES)	16	R/W	0000h	16.4.39/ 720
12FC	TCD Beginning Minor Loop Link, Major Loop Count (Channel Linking Disabled) (DMA_TCD23_BITER_ELINKNO)	16	R/W	0000h	16.4.40/ 721
12FE	TCD Control and Status (DMA_TCD23_CSR)	16	R/W	0000h	16.4.41/ 722
1300	TCD Source Address (DMA_TCD24_SADDR)	32	R/W	0000_0000h	16.4.27/ 712
1304	TCD Transfer Attributes (DMA_TCD24_ATTR)	16	R/W	0000h	16.4.28/ 712
1306	TCD Signed Source Address Offset (DMA_TCD24_SOFF)	16	R/W	0000h	16.4.29/ 713
1308	TCD Minor Byte Count (Minor Loop Mapping Disabled) (DMA_TCD24_NBYTES_MLNO)	32	R/W	0000_0000h	16.4.30/ 713
1308	TCD Signed Minor Loop Offset (Minor Loop Mapping Enabled and Offset Disabled) (DMA_TCD24_NBYTES_MLOFFNO)	32	R/W	0000_0000h	16.4.31/ 714
1308	TCD Signed Minor Loop Offset (Minor Loop Mapping and Offset Enabled) (DMA_TCD24_NBYTES_MLOFFYES)	32	R/W	0000_0000h	16.4.32/ 715
130C	TCD Last Source Address Adjustment (DMA_TCD24_SLAST)	32	R/W	0000_0000h	16.4.33/ 716
1310	TCD Destination Address (DMA_TCD24_DADDR)	32	R/W	0000_0000h	16.4.34/ 717
1314	TCD Current Minor Loop Link, Major Loop Count (Channel Linking Enabled) (DMA_TCD24_CITER_ELINKYES)	16	R/W	0000h	16.4.35/ 717
1314	DMA_TCD24_CITER_ELINKNO	16	R/W	0000h	16.4.36/ 718
1316	TCD Signed Destination Address Offset (DMA_TCD24_DOFF)	16	R/W	0000h	16.4.37/ 719
1318	TCD Last Destination Address Adjustment/Scatter Gather Address (DMA_TCD24_DLASTSGA)	32	R/W	0000_0000h	16.4.38/ 720
131C	TCD Beginning Minor Loop Link, Major Loop Count (Channel Linking Enabled) (DMA_TCD24_BITER_ELINKYES)	16	R/W	0000h	16.4.39/ 720
131C	TCD Beginning Minor Loop Link, Major Loop Count (Channel Linking Disabled) (DMA_TCD24_BITER_ELINKNO)	16	R/W	0000h	16.4.40/ 721
131E	TCD Control and Status (DMA_TCD24_CSR)	16	R/W	0000h	16.4.41/ 722
1320	TCD Source Address (DMA_TCD25_SADDR)	32	R/W	0000_0000h	16.4.27/ 712
1324	TCD Transfer Attributes (DMA_TCD25_ATTR)	16	R/W	0000h	16.4.28/ 712

Table continues on the next page...

DMA memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
1326	TCD Signed Source Address Offset (DMA_TCD25_SOFF)	16	R/W	0000h	16.4.29/ 713
1328	TCD Minor Byte Count (Minor Loop Mapping Disabled) (DMA_TCD25_NBYTES_MLNO)	32	R/W	0000_0000h	16.4.30/ 713
1328	TCD Signed Minor Loop Offset (Minor Loop Mapping Enabled and Offset Disabled) (DMA_TCD25_NBYTES_MLOFFNO)	32	R/W	0000_0000h	16.4.31/ 714
1328	TCD Signed Minor Loop Offset (Minor Loop Mapping and Offset Enabled) (DMA_TCD25_NBYTES_MLOFFYES)	32	R/W	0000_0000h	16.4.32/ 715
132C	TCD Last Source Address Adjustment (DMA_TCD25_SLAST)	32	R/W	0000_0000h	16.4.33/ 716
1330	TCD Destination Address (DMA_TCD25_DADDR)	32	R/W	0000_0000h	16.4.34/ 717
1334	TCD Current Minor Loop Link, Major Loop Count (Channel Linking Enabled) (DMA_TCD25_CITER_ELINKYES)	16	R/W	0000h	16.4.35/ 717
1334	DMA_TCD25_CITER_ELINKNO	16	R/W	0000h	16.4.36/ 718
1336	TCD Signed Destination Address Offset (DMA_TCD25_DOFF)	16	R/W	0000h	16.4.37/ 719
1338	TCD Last Destination Address Adjustment/Scatter Gather Address (DMA_TCD25_DLASTSGA)	32	R/W	0000_0000h	16.4.38/ 720
133C	TCD Beginning Minor Loop Link, Major Loop Count (Channel Linking Enabled) (DMA_TCD25_BITER_ELINKYES)	16	R/W	0000h	16.4.39/ 720
133C	TCD Beginning Minor Loop Link, Major Loop Count (Channel Linking Disabled) (DMA_TCD25_BITER_ELINKNO)	16	R/W	0000h	16.4.40/ 721
133E	TCD Control and Status (DMA_TCD25_CSR)	16	R/W	0000h	16.4.41/ 722
1340	TCD Source Address (DMA_TCD26_SADDR)	32	R/W	0000_0000h	16.4.27/ 712
1344	TCD Transfer Attributes (DMA_TCD26_ATTR)	16	R/W	0000h	16.4.28/ 712
1346	TCD Signed Source Address Offset (DMA_TCD26_SOFF)	16	R/W	0000h	16.4.29/ 713
1348	TCD Minor Byte Count (Minor Loop Mapping Disabled) (DMA_TCD26_NBYTES_MLNO)	32	R/W	0000_0000h	16.4.30/ 713
1348	TCD Signed Minor Loop Offset (Minor Loop Mapping Enabled and Offset Disabled) (DMA_TCD26_NBYTES_MLOFFNO)	32	R/W	0000_0000h	16.4.31/ 714
1348	TCD Signed Minor Loop Offset (Minor Loop Mapping and Offset Enabled) (DMA_TCD26_NBYTES_MLOFFYES)	32	R/W	0000_0000h	16.4.32/ 715
134C	TCD Last Source Address Adjustment (DMA_TCD26_SLAST)	32	R/W	0000_0000h	16.4.33/ 716
1350	TCD Destination Address (DMA_TCD26_DADDR)	32	R/W	0000_0000h	16.4.34/ 717

Table continues on the next page...

DMA memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
1354	TCD Current Minor Loop Link, Major Loop Count (Channel Linking Enabled) (DMA_TCD26_CITER_ELINKYES)	16	R/W	0000h	16.4.35/ 717
1354	DMA_TCD26_CITER_ELINKNO	16	R/W	0000h	16.4.36/ 718
1356	TCD Signed Destination Address Offset (DMA_TCD26_DOFF)	16	R/W	0000h	16.4.37/ 719
1358	TCD Last Destination Address Adjustment/Scatter Gather Address (DMA_TCD26_DLASTSGA)	32	R/W	0000_0000h	16.4.38/ 720
135C	TCD Beginning Minor Loop Link, Major Loop Count (Channel Linking Enabled) (DMA_TCD26_BITER_ELINKYES)	16	R/W	0000h	16.4.39/ 720
135C	TCD Beginning Minor Loop Link, Major Loop Count (Channel Linking Disabled) (DMA_TCD26_BITER_ELINKNO)	16	R/W	0000h	16.4.40/ 721
135E	TCD Control and Status (DMA_TCD26_CSR)	16	R/W	0000h	16.4.41/ 722
1360	TCD Source Address (DMA_TCD27_SADDR)	32	R/W	0000_0000h	16.4.27/ 712
1364	TCD Transfer Attributes (DMA_TCD27_ATTR)	16	R/W	0000h	16.4.28/ 712
1366	TCD Signed Source Address Offset (DMA_TCD27_SOFF)	16	R/W	0000h	16.4.29/ 713
1368	TCD Minor Byte Count (Minor Loop Mapping Disabled) (DMA_TCD27_NBYTES_MLNO)	32	R/W	0000_0000h	16.4.30/ 713
1368	TCD Signed Minor Loop Offset (Minor Loop Mapping Enabled and Offset Disabled) (DMA_TCD27_NBYTES_MLOFFNO)	32	R/W	0000_0000h	16.4.31/ 714
1368	TCD Signed Minor Loop Offset (Minor Loop Mapping and Offset Enabled) (DMA_TCD27_NBYTES_MLOFFYES)	32	R/W	0000_0000h	16.4.32/ 715
136C	TCD Last Source Address Adjustment (DMA_TCD27_SLAST)	32	R/W	0000_0000h	16.4.33/ 716
1370	TCD Destination Address (DMA_TCD27_DADDR)	32	R/W	0000_0000h	16.4.34/ 717
1374	TCD Current Minor Loop Link, Major Loop Count (Channel Linking Enabled) (DMA_TCD27_CITER_ELINKYES)	16	R/W	0000h	16.4.35/ 717
1374	DMA_TCD27_CITER_ELINKNO	16	R/W	0000h	16.4.36/ 718
1376	TCD Signed Destination Address Offset (DMA_TCD27_DOFF)	16	R/W	0000h	16.4.37/ 719
1378	TCD Last Destination Address Adjustment/Scatter Gather Address (DMA_TCD27_DLASTSGA)	32	R/W	0000_0000h	16.4.38/ 720
137C	TCD Beginning Minor Loop Link, Major Loop Count (Channel Linking Enabled) (DMA_TCD27_BITER_ELINKYES)	16	R/W	0000h	16.4.39/ 720

Table continues on the next page...

DMA memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
137C	TCD Beginning Minor Loop Link, Major Loop Count (Channel Linking Disabled) (DMA_TCD27_BITER_ELINKNO)	16	R/W	0000h	16.4.40/ 721
137E	TCD Control and Status (DMA_TCD27_CSR)	16	R/W	0000h	16.4.41/ 722
1380	TCD Source Address (DMA_TCD28_SADDR)	32	R/W	0000_0000h	16.4.27/ 712
1384	TCD Transfer Attributes (DMA_TCD28_ATTR)	16	R/W	0000h	16.4.28/ 712
1386	TCD Signed Source Address Offset (DMA_TCD28_SOFF)	16	R/W	0000h	16.4.29/ 713
1388	TCD Minor Byte Count (Minor Loop Mapping Disabled) (DMA_TCD28_NBYTES_MLNO)	32	R/W	0000_0000h	16.4.30/ 713
1388	TCD Signed Minor Loop Offset (Minor Loop Mapping Enabled and Offset Disabled) (DMA_TCD28_NBYTES_MLOFFNO)	32	R/W	0000_0000h	16.4.31/ 714
1388	TCD Signed Minor Loop Offset (Minor Loop Mapping and Offset Enabled) (DMA_TCD28_NBYTES_MLOFFYES)	32	R/W	0000_0000h	16.4.32/ 715
138C	TCD Last Source Address Adjustment (DMA_TCD28_SLAST)	32	R/W	0000_0000h	16.4.33/ 716
1390	TCD Destination Address (DMA_TCD28_DADDR)	32	R/W	0000_0000h	16.4.34/ 717
1394	TCD Current Minor Loop Link, Major Loop Count (Channel Linking Enabled) (DMA_TCD28_CITER_ELINKYES)	16	R/W	0000h	16.4.35/ 717
1394	DMA_TCD28_CITER_ELINKNO	16	R/W	0000h	16.4.36/ 718
1396	TCD Signed Destination Address Offset (DMA_TCD28_DOFF)	16	R/W	0000h	16.4.37/ 719
1398	TCD Last Destination Address Adjustment/Scatter Gather Address (DMA_TCD28_DLASTSGA)	32	R/W	0000_0000h	16.4.38/ 720
139C	TCD Beginning Minor Loop Link, Major Loop Count (Channel Linking Enabled) (DMA_TCD28_BITER_ELINKYES)	16	R/W	0000h	16.4.39/ 720
139C	TCD Beginning Minor Loop Link, Major Loop Count (Channel Linking Disabled) (DMA_TCD28_BITER_ELINKNO)	16	R/W	0000h	16.4.40/ 721
139E	TCD Control and Status (DMA_TCD28_CSR)	16	R/W	0000h	16.4.41/ 722
13A0	TCD Source Address (DMA_TCD29_SADDR)	32	R/W	0000_0000h	16.4.27/ 712
13A4	TCD Transfer Attributes (DMA_TCD29_ATTR)	16	R/W	0000h	16.4.28/ 712
13A6	TCD Signed Source Address Offset (DMA_TCD29_SOFF)	16	R/W	0000h	16.4.29/ 713
13A8	TCD Minor Byte Count (Minor Loop Mapping Disabled) (DMA_TCD29_NBYTES_MLNO)	32	R/W	0000_0000h	16.4.30/ 713

Table continues on the next page...

DMA memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
13A8	TCD Signed Minor Loop Offset (Minor Loop Mapping Enabled and Offset Disabled) (DMA_TCD29_NBYTES_MLOFFNO)	32	R/W	0000_0000h	16.4.31/ 714
13A8	TCD Signed Minor Loop Offset (Minor Loop Mapping and Offset Enabled) (DMA_TCD29_NBYTES_MLOFFYES)	32	R/W	0000_0000h	16.4.32/ 715
13AC	TCD Last Source Address Adjustment (DMA_TCD29_SLAST)	32	R/W	0000_0000h	16.4.33/ 716
13B0	TCD Destination Address (DMA_TCD29_DADDR)	32	R/W	0000_0000h	16.4.34/ 717
13B4	TCD Current Minor Loop Link, Major Loop Count (Channel Linking Enabled) (DMA_TCD29_CITER_ELINKYES)	16	R/W	0000h	16.4.35/ 717
13B4	DMA_TCD29_CITER_ELINKNO	16	R/W	0000h	16.4.36/ 718
13B6	TCD Signed Destination Address Offset (DMA_TCD29_DOFF)	16	R/W	0000h	16.4.37/ 719
13B8	TCD Last Destination Address Adjustment/Scatter Gather Address (DMA_TCD29_DLASTSGA)	32	R/W	0000_0000h	16.4.38/ 720
13BC	TCD Beginning Minor Loop Link, Major Loop Count (Channel Linking Enabled) (DMA_TCD29_BITER_ELINKYES)	16	R/W	0000h	16.4.39/ 720
13BC	TCD Beginning Minor Loop Link, Major Loop Count (Channel Linking Disabled) (DMA_TCD29_BITER_ELINKNO)	16	R/W	0000h	16.4.40/ 721
13BE	TCD Control and Status (DMA_TCD29_CSR)	16	R/W	0000h	16.4.41/ 722
13C0	TCD Source Address (DMA_TCD30_SADDR)	32	R/W	0000_0000h	16.4.27/ 712
13C4	TCD Transfer Attributes (DMA_TCD30_ATTR)	16	R/W	0000h	16.4.28/ 712
13C6	TCD Signed Source Address Offset (DMA_TCD30_SOFF)	16	R/W	0000h	16.4.29/ 713
13C8	TCD Minor Byte Count (Minor Loop Mapping Disabled) (DMA_TCD30_NBYTES_MLNO)	32	R/W	0000_0000h	16.4.30/ 713
13C8	TCD Signed Minor Loop Offset (Minor Loop Mapping Enabled and Offset Disabled) (DMA_TCD30_NBYTES_MLOFFNO)	32	R/W	0000_0000h	16.4.31/ 714
13C8	TCD Signed Minor Loop Offset (Minor Loop Mapping and Offset Enabled) (DMA_TCD30_NBYTES_MLOFFYES)	32	R/W	0000_0000h	16.4.32/ 715
13CC	TCD Last Source Address Adjustment (DMA_TCD30_SLAST)	32	R/W	0000_0000h	16.4.33/ 716
13D0	TCD Destination Address (DMA_TCD30_DADDR)	32	R/W	0000_0000h	16.4.34/ 717
13D4	TCD Current Minor Loop Link, Major Loop Count (Channel Linking Enabled) (DMA_TCD30_CITER_ELINKYES)	16	R/W	0000h	16.4.35/ 717
13D4	DMA_TCD30_CITER_ELINKNO	16	R/W	0000h	16.4.36/ 718

Table continues on the next page...

DMA memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
13D6	TCD Signed Destination Address Offset (DMA_TCD30_DOFF)	16	R/W	0000h	16.4.37/ 719
13D8	TCD Last Destination Address Adjustment/Scatter Gather Address (DMA_TCD30_DLASTSGA)	32	R/W	0000_0000h	16.4.38/ 720
13DC	TCD Beginning Minor Loop Link, Major Loop Count (Channel Linking Enabled) (DMA_TCD30_BITER_ELINKYES)	16	R/W	0000h	16.4.39/ 720
13DC	TCD Beginning Minor Loop Link, Major Loop Count (Channel Linking Disabled) (DMA_TCD30_BITER_ELINKNO)	16	R/W	0000h	16.4.40/ 721
13DE	TCD Control and Status (DMA_TCD30_CSR)	16	R/W	0000h	16.4.41/ 722
13E0	TCD Source Address (DMA_TCD31_SADDR)	32	R/W	0000_0000h	16.4.27/ 712
13E4	TCD Transfer Attributes (DMA_TCD31_ATTR)	16	R/W	0000h	16.4.28/ 712
13E6	TCD Signed Source Address Offset (DMA_TCD31_SOFF)	16	R/W	0000h	16.4.29/ 713
13E8	TCD Minor Byte Count (Minor Loop Mapping Disabled) (DMA_TCD31_NBYTES_MLNO)	32	R/W	0000_0000h	16.4.30/ 713
13E8	TCD Signed Minor Loop Offset (Minor Loop Mapping Enabled and Offset Disabled) (DMA_TCD31_NBYTES_MLOFFNO)	32	R/W	0000_0000h	16.4.31/ 714
13E8	TCD Signed Minor Loop Offset (Minor Loop Mapping and Offset Enabled) (DMA_TCD31_NBYTES_MLOFFYES)	32	R/W	0000_0000h	16.4.32/ 715
13EC	TCD Last Source Address Adjustment (DMA_TCD31_SLAST)	32	R/W	0000_0000h	16.4.33/ 716
13F0	TCD Destination Address (DMA_TCD31_DADDR)	32	R/W	0000_0000h	16.4.34/ 717
13F4	TCD Current Minor Loop Link, Major Loop Count (Channel Linking Enabled) (DMA_TCD31_CITER_ELINKYES)	16	R/W	0000h	16.4.35/ 717
13F4	DMA_TCD31_CITER_ELINKNO	16	R/W	0000h	16.4.36/ 718
13F6	TCD Signed Destination Address Offset (DMA_TCD31_DOFF)	16	R/W	0000h	16.4.37/ 719
13F8	TCD Last Destination Address Adjustment/Scatter Gather Address (DMA_TCD31_DLASTSGA)	32	R/W	0000_0000h	16.4.38/ 720
13FC	TCD Beginning Minor Loop Link, Major Loop Count (Channel Linking Enabled) (DMA_TCD31_BITER_ELINKYES)	16	R/W	0000h	16.4.39/ 720
13FC	TCD Beginning Minor Loop Link, Major Loop Count (Channel Linking Disabled) (DMA_TCD31_BITER_ELINKNO)	16	R/W	0000h	16.4.40/ 721
13FE	TCD Control and Status (DMA_TCD31_CSR)	16	R/W	0000h	16.4.41/ 722

Table continues on the next page...

DMA memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
1400	TCD Source Address (DMA_TCD32_SADDR)	32	R/W	0000_0000h	16.4.27/ 712
1404	TCD Transfer Attributes (DMA_TCD32_ATTR)	16	R/W	0000h	16.4.28/ 712
1406	TCD Signed Source Address Offset (DMA_TCD32_SOFF)	16	R/W	0000h	16.4.29/ 713
1408	TCD Minor Byte Count (Minor Loop Mapping Disabled) (DMA_TCD32_NBYTES_MLNO)	32	R/W	0000_0000h	16.4.30/ 713
1408	TCD Signed Minor Loop Offset (Minor Loop Mapping Enabled and Offset Disabled) (DMA_TCD32_NBYTES_MLOFFNO)	32	R/W	0000_0000h	16.4.31/ 714
1408	TCD Signed Minor Loop Offset (Minor Loop Mapping and Offset Enabled) (DMA_TCD32_NBYTES_MLOFFYES)	32	R/W	0000_0000h	16.4.32/ 715
140C	TCD Last Source Address Adjustment (DMA_TCD32_SLAST)	32	R/W	0000_0000h	16.4.33/ 716
1410	TCD Destination Address (DMA_TCD32_DADDR)	32	R/W	0000_0000h	16.4.34/ 717
1414	TCD Current Minor Loop Link, Major Loop Count (Channel Linking Enabled) (DMA_TCD32_CITER_ELINKYES)	16	R/W	0000h	16.4.35/ 717
1414	DMA_TCD32_CITER_ELINKNO	16	R/W	0000h	16.4.36/ 718
1416	TCD Signed Destination Address Offset (DMA_TCD32_DOFF)	16	R/W	0000h	16.4.37/ 719
1418	TCD Last Destination Address Adjustment/Scatter Gather Address (DMA_TCD32_DLASTSGA)	32	R/W	0000_0000h	16.4.38/ 720
141C	TCD Beginning Minor Loop Link, Major Loop Count (Channel Linking Enabled) (DMA_TCD32_BITER_ELINKYES)	16	R/W	0000h	16.4.39/ 720
141C	TCD Beginning Minor Loop Link, Major Loop Count (Channel Linking Disabled) (DMA_TCD32_BITER_ELINKNO)	16	R/W	0000h	16.4.40/ 721
141E	TCD Control and Status (DMA_TCD32_CSR)	16	R/W	0000h	16.4.41/ 722
1420	TCD Source Address (DMA_TCD33_SADDR)	32	R/W	0000_0000h	16.4.27/ 712
1424	TCD Transfer Attributes (DMA_TCD33_ATTR)	16	R/W	0000h	16.4.28/ 712
1426	TCD Signed Source Address Offset (DMA_TCD33_SOFF)	16	R/W	0000h	16.4.29/ 713
1428	TCD Minor Byte Count (Minor Loop Mapping Disabled) (DMA_TCD33_NBYTES_MLNO)	32	R/W	0000_0000h	16.4.30/ 713
1428	TCD Signed Minor Loop Offset (Minor Loop Mapping Enabled and Offset Disabled) (DMA_TCD33_NBYTES_MLOFFNO)	32	R/W	0000_0000h	16.4.31/ 714
1428	TCD Signed Minor Loop Offset (Minor Loop Mapping and Offset Enabled) (DMA_TCD33_NBYTES_MLOFFYES)	32	R/W	0000_0000h	16.4.32/ 715

Table continues on the next page...

DMA memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
142C	TCD Last Source Address Adjustment (DMA_TCD33_SLAST)	32	R/W	0000_0000h	16.4.33/ 716
1430	TCD Destination Address (DMA_TCD33_DADDR)	32	R/W	0000_0000h	16.4.34/ 717
1434	TCD Current Minor Loop Link, Major Loop Count (Channel Linking Enabled) (DMA_TCD33_CITER_ELINKYES)	16	R/W	0000h	16.4.35/ 717
1434	DMA_TCD33_CITER_ELINKNO	16	R/W	0000h	16.4.36/ 718
1436	TCD Signed Destination Address Offset (DMA_TCD33_DOFF)	16	R/W	0000h	16.4.37/ 719
1438	TCD Last Destination Address Adjustment/Scatter Gather Address (DMA_TCD33_DLASTSGA)	32	R/W	0000_0000h	16.4.38/ 720
143C	TCD Beginning Minor Loop Link, Major Loop Count (Channel Linking Enabled) (DMA_TCD33_BITER_ELINKYES)	16	R/W	0000h	16.4.39/ 720
143C	TCD Beginning Minor Loop Link, Major Loop Count (Channel Linking Disabled) (DMA_TCD33_BITER_ELINKNO)	16	R/W	0000h	16.4.40/ 721
143E	TCD Control and Status (DMA_TCD33_CSR)	16	R/W	0000h	16.4.41/ 722
1440	TCD Source Address (DMA_TCD34_SADDR)	32	R/W	0000_0000h	16.4.27/ 712
1444	TCD Transfer Attributes (DMA_TCD34_ATTR)	16	R/W	0000h	16.4.28/ 712
1446	TCD Signed Source Address Offset (DMA_TCD34_SOFF)	16	R/W	0000h	16.4.29/ 713
1448	TCD Minor Byte Count (Minor Loop Mapping Disabled) (DMA_TCD34_NBYTES_MLNO)	32	R/W	0000_0000h	16.4.30/ 713
1448	TCD Signed Minor Loop Offset (Minor Loop Mapping Enabled and Offset Disabled) (DMA_TCD34_NBYTES_MLOFFNO)	32	R/W	0000_0000h	16.4.31/ 714
1448	TCD Signed Minor Loop Offset (Minor Loop Mapping and Offset Enabled) (DMA_TCD34_NBYTES_MLOFFYES)	32	R/W	0000_0000h	16.4.32/ 715
144C	TCD Last Source Address Adjustment (DMA_TCD34_SLAST)	32	R/W	0000_0000h	16.4.33/ 716
1450	TCD Destination Address (DMA_TCD34_DADDR)	32	R/W	0000_0000h	16.4.34/ 717
1454	TCD Current Minor Loop Link, Major Loop Count (Channel Linking Enabled) (DMA_TCD34_CITER_ELINKYES)	16	R/W	0000h	16.4.35/ 717
1454	DMA_TCD34_CITER_ELINKNO	16	R/W	0000h	16.4.36/ 718
1456	TCD Signed Destination Address Offset (DMA_TCD34_DOFF)	16	R/W	0000h	16.4.37/ 719
1458	TCD Last Destination Address Adjustment/Scatter Gather Address (DMA_TCD34_DLASTSGA)	32	R/W	0000_0000h	16.4.38/ 720

Table continues on the next page...

DMA memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
145C	TCD Beginning Minor Loop Link, Major Loop Count (Channel Linking Enabled) (DMA_TCD34_BITER_ELINKYES)	16	R/W	0000h	16.4.39/ 720
145C	TCD Beginning Minor Loop Link, Major Loop Count (Channel Linking Disabled) (DMA_TCD34_BITER_ELINKNO)	16	R/W	0000h	16.4.40/ 721
145E	TCD Control and Status (DMA_TCD34_CSR)	16	R/W	0000h	16.4.41/ 722
1460	TCD Source Address (DMA_TCD35_SADDR)	32	R/W	0000_0000h	16.4.27/ 712
1464	TCD Transfer Attributes (DMA_TCD35_ATTR)	16	R/W	0000h	16.4.28/ 712
1466	TCD Signed Source Address Offset (DMA_TCD35_SOFF)	16	R/W	0000h	16.4.29/ 713
1468	TCD Minor Byte Count (Minor Loop Mapping Disabled) (DMA_TCD35_NBYTES_MLNO)	32	R/W	0000_0000h	16.4.30/ 713
1468	TCD Signed Minor Loop Offset (Minor Loop Mapping Enabled and Offset Disabled) (DMA_TCD35_NBYTES_MLOFFNO)	32	R/W	0000_0000h	16.4.31/ 714
1468	TCD Signed Minor Loop Offset (Minor Loop Mapping and Offset Enabled) (DMA_TCD35_NBYTES_MLOFFYES)	32	R/W	0000_0000h	16.4.32/ 715
146C	TCD Last Source Address Adjustment (DMA_TCD35_SLAST)	32	R/W	0000_0000h	16.4.33/ 716
1470	TCD Destination Address (DMA_TCD35_DADDR)	32	R/W	0000_0000h	16.4.34/ 717
1474	TCD Current Minor Loop Link, Major Loop Count (Channel Linking Enabled) (DMA_TCD35_CITER_ELINKYES)	16	R/W	0000h	16.4.35/ 717
1474	DMA_TCD35_CITER_ELINKNO	16	R/W	0000h	16.4.36/ 718
1476	TCD Signed Destination Address Offset (DMA_TCD35_DOFF)	16	R/W	0000h	16.4.37/ 719
1478	TCD Last Destination Address Adjustment/Scatter Gather Address (DMA_TCD35_DLASTSGA)	32	R/W	0000_0000h	16.4.38/ 720
147C	TCD Beginning Minor Loop Link, Major Loop Count (Channel Linking Enabled) (DMA_TCD35_BITER_ELINKYES)	16	R/W	0000h	16.4.39/ 720
147C	TCD Beginning Minor Loop Link, Major Loop Count (Channel Linking Disabled) (DMA_TCD35_BITER_ELINKNO)	16	R/W	0000h	16.4.40/ 721
147E	TCD Control and Status (DMA_TCD35_CSR)	16	R/W	0000h	16.4.41/ 722
1480	TCD Source Address (DMA_TCD36_SADDR)	32	R/W	0000_0000h	16.4.27/ 712
1484	TCD Transfer Attributes (DMA_TCD36_ATTR)	16	R/W	0000h	16.4.28/ 712

Table continues on the next page...

DMA memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
1486	TCD Signed Source Address Offset (DMA_TCD36_SOFF)	16	R/W	0000h	16.4.29/ 713
1488	TCD Minor Byte Count (Minor Loop Mapping Disabled) (DMA_TCD36_NBYTES_MLNO)	32	R/W	0000_0000h	16.4.30/ 713
1488	TCD Signed Minor Loop Offset (Minor Loop Mapping Enabled and Offset Disabled) (DMA_TCD36_NBYTES_MLOFFNO)	32	R/W	0000_0000h	16.4.31/ 714
1488	TCD Signed Minor Loop Offset (Minor Loop Mapping and Offset Enabled) (DMA_TCD36_NBYTES_MLOFFYES)	32	R/W	0000_0000h	16.4.32/ 715
148C	TCD Last Source Address Adjustment (DMA_TCD36_SLAST)	32	R/W	0000_0000h	16.4.33/ 716
1490	TCD Destination Address (DMA_TCD36_DADDR)	32	R/W	0000_0000h	16.4.34/ 717
1494	TCD Current Minor Loop Link, Major Loop Count (Channel Linking Enabled) (DMA_TCD36_CITER_ELINKYES)	16	R/W	0000h	16.4.35/ 717
1494	DMA_TCD36_CITER_ELINKNO	16	R/W	0000h	16.4.36/ 718
1496	TCD Signed Destination Address Offset (DMA_TCD36_DOFF)	16	R/W	0000h	16.4.37/ 719
1498	TCD Last Destination Address Adjustment/Scatter Gather Address (DMA_TCD36_DLASTSGA)	32	R/W	0000_0000h	16.4.38/ 720
149C	TCD Beginning Minor Loop Link, Major Loop Count (Channel Linking Enabled) (DMA_TCD36_BITER_ELINKYES)	16	R/W	0000h	16.4.39/ 720
149C	TCD Beginning Minor Loop Link, Major Loop Count (Channel Linking Disabled) (DMA_TCD36_BITER_ELINKNO)	16	R/W	0000h	16.4.40/ 721
149E	TCD Control and Status (DMA_TCD36_CSR)	16	R/W	0000h	16.4.41/ 722
14A0	TCD Source Address (DMA_TCD37_SADDR)	32	R/W	0000_0000h	16.4.27/ 712
14A4	TCD Transfer Attributes (DMA_TCD37_ATTR)	16	R/W	0000h	16.4.28/ 712
14A6	TCD Signed Source Address Offset (DMA_TCD37_SOFF)	16	R/W	0000h	16.4.29/ 713
14A8	TCD Minor Byte Count (Minor Loop Mapping Disabled) (DMA_TCD37_NBYTES_MLNO)	32	R/W	0000_0000h	16.4.30/ 713
14A8	TCD Signed Minor Loop Offset (Minor Loop Mapping Enabled and Offset Disabled) (DMA_TCD37_NBYTES_MLOFFNO)	32	R/W	0000_0000h	16.4.31/ 714
14A8	TCD Signed Minor Loop Offset (Minor Loop Mapping and Offset Enabled) (DMA_TCD37_NBYTES_MLOFFYES)	32	R/W	0000_0000h	16.4.32/ 715
14AC	TCD Last Source Address Adjustment (DMA_TCD37_SLAST)	32	R/W	0000_0000h	16.4.33/ 716
14B0	TCD Destination Address (DMA_TCD37_DADDR)	32	R/W	0000_0000h	16.4.34/ 717

Table continues on the next page...

DMA memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
14B4	TCD Current Minor Loop Link, Major Loop Count (Channel Linking Enabled) (DMA_TCD37_CITER_ELINKYES)	16	R/W	0000h	16.4.35/ 717
14B4	DMA_TCD37_CITER_ELINKNO	16	R/W	0000h	16.4.36/ 718
14B6	TCD Signed Destination Address Offset (DMA_TCD37_DOFF)	16	R/W	0000h	16.4.37/ 719
14B8	TCD Last Destination Address Adjustment/Scatter Gather Address (DMA_TCD37_DLASTSGA)	32	R/W	0000_0000h	16.4.38/ 720
14BC	TCD Beginning Minor Loop Link, Major Loop Count (Channel Linking Enabled) (DMA_TCD37_BITER_ELINKYES)	16	R/W	0000h	16.4.39/ 720
14BC	TCD Beginning Minor Loop Link, Major Loop Count (Channel Linking Disabled) (DMA_TCD37_BITER_ELINKNO)	16	R/W	0000h	16.4.40/ 721
14BE	TCD Control and Status (DMA_TCD37_CSR)	16	R/W	0000h	16.4.41/ 722
14C0	TCD Source Address (DMA_TCD38_SADDR)	32	R/W	0000_0000h	16.4.27/ 712
14C4	TCD Transfer Attributes (DMA_TCD38_ATTR)	16	R/W	0000h	16.4.28/ 712
14C6	TCD Signed Source Address Offset (DMA_TCD38_SOFF)	16	R/W	0000h	16.4.29/ 713
14C8	TCD Minor Byte Count (Minor Loop Mapping Disabled) (DMA_TCD38_NBYTES_MLNO)	32	R/W	0000_0000h	16.4.30/ 713
14C8	TCD Signed Minor Loop Offset (Minor Loop Mapping Enabled and Offset Disabled) (DMA_TCD38_NBYTES_MLOFFNO)	32	R/W	0000_0000h	16.4.31/ 714
14C8	TCD Signed Minor Loop Offset (Minor Loop Mapping and Offset Enabled) (DMA_TCD38_NBYTES_MLOFFYES)	32	R/W	0000_0000h	16.4.32/ 715
14CC	TCD Last Source Address Adjustment (DMA_TCD38_SLAST)	32	R/W	0000_0000h	16.4.33/ 716
14D0	TCD Destination Address (DMA_TCD38_DADDR)	32	R/W	0000_0000h	16.4.34/ 717
14D4	TCD Current Minor Loop Link, Major Loop Count (Channel Linking Enabled) (DMA_TCD38_CITER_ELINKYES)	16	R/W	0000h	16.4.35/ 717
14D4	DMA_TCD38_CITER_ELINKNO	16	R/W	0000h	16.4.36/ 718
14D6	TCD Signed Destination Address Offset (DMA_TCD38_DOFF)	16	R/W	0000h	16.4.37/ 719
14D8	TCD Last Destination Address Adjustment/Scatter Gather Address (DMA_TCD38_DLASTSGA)	32	R/W	0000_0000h	16.4.38/ 720
14DC	TCD Beginning Minor Loop Link, Major Loop Count (Channel Linking Enabled) (DMA_TCD38_BITER_ELINKYES)	16	R/W	0000h	16.4.39/ 720

Table continues on the next page...

DMA memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
14DC	TCD Beginning Minor Loop Link, Major Loop Count (Channel Linking Disabled) (DMA_TCD38_BITER_ELINKNO)	16	R/W	0000h	16.4.40/ 721
14DE	TCD Control and Status (DMA_TCD38_CSR)	16	R/W	0000h	16.4.41/ 722
14E0	TCD Source Address (DMA_TCD39_SADDR)	32	R/W	0000_0000h	16.4.27/ 712
14E4	TCD Transfer Attributes (DMA_TCD39_ATTR)	16	R/W	0000h	16.4.28/ 712
14E6	TCD Signed Source Address Offset (DMA_TCD39_SOFF)	16	R/W	0000h	16.4.29/ 713
14E8	TCD Minor Byte Count (Minor Loop Mapping Disabled) (DMA_TCD39_NBYTES_MLNO)	32	R/W	0000_0000h	16.4.30/ 713
14E8	TCD Signed Minor Loop Offset (Minor Loop Mapping Enabled and Offset Disabled) (DMA_TCD39_NBYTES_MLOFFNO)	32	R/W	0000_0000h	16.4.31/ 714
14E8	TCD Signed Minor Loop Offset (Minor Loop Mapping and Offset Enabled) (DMA_TCD39_NBYTES_MLOFFYES)	32	R/W	0000_0000h	16.4.32/ 715
14EC	TCD Last Source Address Adjustment (DMA_TCD39_SLAST)	32	R/W	0000_0000h	16.4.33/ 716
14F0	TCD Destination Address (DMA_TCD39_DADDR)	32	R/W	0000_0000h	16.4.34/ 717
14F4	TCD Current Minor Loop Link, Major Loop Count (Channel Linking Enabled) (DMA_TCD39_CITER_ELINKYES)	16	R/W	0000h	16.4.35/ 717
14F4	DMA_TCD39_CITER_ELINKNO	16	R/W	0000h	16.4.36/ 718
14F6	TCD Signed Destination Address Offset (DMA_TCD39_DOFF)	16	R/W	0000h	16.4.37/ 719
14F8	TCD Last Destination Address Adjustment/Scatter Gather Address (DMA_TCD39_DLASTSGA)	32	R/W	0000_0000h	16.4.38/ 720
14FC	TCD Beginning Minor Loop Link, Major Loop Count (Channel Linking Enabled) (DMA_TCD39_BITER_ELINKYES)	16	R/W	0000h	16.4.39/ 720
14FC	TCD Beginning Minor Loop Link, Major Loop Count (Channel Linking Disabled) (DMA_TCD39_BITER_ELINKNO)	16	R/W	0000h	16.4.40/ 721
14FE	TCD Control and Status (DMA_TCD39_CSR)	16	R/W	0000h	16.4.41/ 722
1500	TCD Source Address (DMA_TCD40_SADDR)	32	R/W	0000_0000h	16.4.27/ 712
1504	TCD Transfer Attributes (DMA_TCD40_ATTR)	16	R/W	0000h	16.4.28/ 712
1506	TCD Signed Source Address Offset (DMA_TCD40_SOFF)	16	R/W	0000h	16.4.29/ 713
1508	TCD Minor Byte Count (Minor Loop Mapping Disabled) (DMA_TCD40_NBYTES_MLNO)	32	R/W	0000_0000h	16.4.30/ 713

Table continues on the next page...

DMA memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
1508	TCD Signed Minor Loop Offset (Minor Loop Mapping Enabled and Offset Disabled) (DMA_TCD40_NBYTES_MLOFFNO)	32	R/W	0000_0000h	16.4.31/ 714
1508	TCD Signed Minor Loop Offset (Minor Loop Mapping and Offset Enabled) (DMA_TCD40_NBYTES_MLOFFYES)	32	R/W	0000_0000h	16.4.32/ 715
150C	TCD Last Source Address Adjustment (DMA_TCD40_SLAST)	32	R/W	0000_0000h	16.4.33/ 716
1510	TCD Destination Address (DMA_TCD40_DADDR)	32	R/W	0000_0000h	16.4.34/ 717
1514	TCD Current Minor Loop Link, Major Loop Count (Channel Linking Enabled) (DMA_TCD40_CITER_ELINKYES)	16	R/W	0000h	16.4.35/ 717
1514	DMA_TCD40_CITER_ELINKNO	16	R/W	0000h	16.4.36/ 718
1516	TCD Signed Destination Address Offset (DMA_TCD40_DOFF)	16	R/W	0000h	16.4.37/ 719
1518	TCD Last Destination Address Adjustment/Scatter Gather Address (DMA_TCD40_DLASTSGA)	32	R/W	0000_0000h	16.4.38/ 720
151C	TCD Beginning Minor Loop Link, Major Loop Count (Channel Linking Enabled) (DMA_TCD40_BITER_ELINKYES)	16	R/W	0000h	16.4.39/ 720
151C	TCD Beginning Minor Loop Link, Major Loop Count (Channel Linking Disabled) (DMA_TCD40_BITER_ELINKNO)	16	R/W	0000h	16.4.40/ 721
151E	TCD Control and Status (DMA_TCD40_CSR)	16	R/W	0000h	16.4.41/ 722
1520	TCD Source Address (DMA_TCD41_SADDR)	32	R/W	0000_0000h	16.4.27/ 712
1524	TCD Transfer Attributes (DMA_TCD41_ATTR)	16	R/W	0000h	16.4.28/ 712
1526	TCD Signed Source Address Offset (DMA_TCD41_SOFF)	16	R/W	0000h	16.4.29/ 713
1528	TCD Minor Byte Count (Minor Loop Mapping Disabled) (DMA_TCD41_NBYTES_MLNO)	32	R/W	0000_0000h	16.4.30/ 713
1528	TCD Signed Minor Loop Offset (Minor Loop Mapping Enabled and Offset Disabled) (DMA_TCD41_NBYTES_MLOFFNO)	32	R/W	0000_0000h	16.4.31/ 714
1528	TCD Signed Minor Loop Offset (Minor Loop Mapping and Offset Enabled) (DMA_TCD41_NBYTES_MLOFFYES)	32	R/W	0000_0000h	16.4.32/ 715
152C	TCD Last Source Address Adjustment (DMA_TCD41_SLAST)	32	R/W	0000_0000h	16.4.33/ 716
1530	TCD Destination Address (DMA_TCD41_DADDR)	32	R/W	0000_0000h	16.4.34/ 717
1534	TCD Current Minor Loop Link, Major Loop Count (Channel Linking Enabled) (DMA_TCD41_CITER_ELINKYES)	16	R/W	0000h	16.4.35/ 717
1534	DMA_TCD41_CITER_ELINKNO	16	R/W	0000h	16.4.36/ 718

Table continues on the next page...

DMA memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
1536	TCD Signed Destination Address Offset (DMA_TCD41_DOFF)	16	R/W	0000h	16.4.37/ 719
1538	TCD Last Destination Address Adjustment/Scatter Gather Address (DMA_TCD41_DLASTSGA)	32	R/W	0000_0000h	16.4.38/ 720
153C	TCD Beginning Minor Loop Link, Major Loop Count (Channel Linking Enabled) (DMA_TCD41_BITER_ELINKYES)	16	R/W	0000h	16.4.39/ 720
153C	TCD Beginning Minor Loop Link, Major Loop Count (Channel Linking Disabled) (DMA_TCD41_BITER_ELINKNO)	16	R/W	0000h	16.4.40/ 721
153E	TCD Control and Status (DMA_TCD41_CSR)	16	R/W	0000h	16.4.41/ 722
1540	TCD Source Address (DMA_TCD42_SADDR)	32	R/W	0000_0000h	16.4.27/ 712
1544	TCD Transfer Attributes (DMA_TCD42_ATTR)	16	R/W	0000h	16.4.28/ 712
1546	TCD Signed Source Address Offset (DMA_TCD42_SOFF)	16	R/W	0000h	16.4.29/ 713
1548	TCD Minor Byte Count (Minor Loop Mapping Disabled) (DMA_TCD42_NBYTES_MLNO)	32	R/W	0000_0000h	16.4.30/ 713
1548	TCD Signed Minor Loop Offset (Minor Loop Mapping Enabled and Offset Disabled) (DMA_TCD42_NBYTES_MLOFFNO)	32	R/W	0000_0000h	16.4.31/ 714
1548	TCD Signed Minor Loop Offset (Minor Loop Mapping and Offset Enabled) (DMA_TCD42_NBYTES_MLOFFYES)	32	R/W	0000_0000h	16.4.32/ 715
154C	TCD Last Source Address Adjustment (DMA_TCD42_SLAST)	32	R/W	0000_0000h	16.4.33/ 716
1550	TCD Destination Address (DMA_TCD42_DADDR)	32	R/W	0000_0000h	16.4.34/ 717
1554	TCD Current Minor Loop Link, Major Loop Count (Channel Linking Enabled) (DMA_TCD42_CITER_ELINKYES)	16	R/W	0000h	16.4.35/ 717
1554	DMA_TCD42_CITER_ELINKNO	16	R/W	0000h	16.4.36/ 718
1556	TCD Signed Destination Address Offset (DMA_TCD42_DOFF)	16	R/W	0000h	16.4.37/ 719
1558	TCD Last Destination Address Adjustment/Scatter Gather Address (DMA_TCD42_DLASTSGA)	32	R/W	0000_0000h	16.4.38/ 720
155C	TCD Beginning Minor Loop Link, Major Loop Count (Channel Linking Enabled) (DMA_TCD42_BITER_ELINKYES)	16	R/W	0000h	16.4.39/ 720
155C	TCD Beginning Minor Loop Link, Major Loop Count (Channel Linking Disabled) (DMA_TCD42_BITER_ELINKNO)	16	R/W	0000h	16.4.40/ 721
155E	TCD Control and Status (DMA_TCD42_CSR)	16	R/W	0000h	16.4.41/ 722

Table continues on the next page...

DMA memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
1560	TCD Source Address (DMA_TCD43_SADDR)	32	R/W	0000_0000h	16.4.27/ 712
1564	TCD Transfer Attributes (DMA_TCD43_ATTR)	16	R/W	0000h	16.4.28/ 712
1566	TCD Signed Source Address Offset (DMA_TCD43_SOFF)	16	R/W	0000h	16.4.29/ 713
1568	TCD Minor Byte Count (Minor Loop Mapping Disabled) (DMA_TCD43_NBYTES_MLNO)	32	R/W	0000_0000h	16.4.30/ 713
1568	TCD Signed Minor Loop Offset (Minor Loop Mapping Enabled and Offset Disabled) (DMA_TCD43_NBYTES_MLOFFNO)	32	R/W	0000_0000h	16.4.31/ 714
1568	TCD Signed Minor Loop Offset (Minor Loop Mapping and Offset Enabled) (DMA_TCD43_NBYTES_MLOFFYES)	32	R/W	0000_0000h	16.4.32/ 715
156C	TCD Last Source Address Adjustment (DMA_TCD43_SLAST)	32	R/W	0000_0000h	16.4.33/ 716
1570	TCD Destination Address (DMA_TCD43_DADDR)	32	R/W	0000_0000h	16.4.34/ 717
1574	TCD Current Minor Loop Link, Major Loop Count (Channel Linking Enabled) (DMA_TCD43_CITER_ELINKYES)	16	R/W	0000h	16.4.35/ 717
1574	DMA_TCD43_CITER_ELINKNO	16	R/W	0000h	16.4.36/ 718
1576	TCD Signed Destination Address Offset (DMA_TCD43_DOFF)	16	R/W	0000h	16.4.37/ 719
1578	TCD Last Destination Address Adjustment/Scatter Gather Address (DMA_TCD43_DLASTSGA)	32	R/W	0000_0000h	16.4.38/ 720
157C	TCD Beginning Minor Loop Link, Major Loop Count (Channel Linking Enabled) (DMA_TCD43_BITER_ELINKYES)	16	R/W	0000h	16.4.39/ 720
157C	TCD Beginning Minor Loop Link, Major Loop Count (Channel Linking Disabled) (DMA_TCD43_BITER_ELINKNO)	16	R/W	0000h	16.4.40/ 721
157E	TCD Control and Status (DMA_TCD43_CSR)	16	R/W	0000h	16.4.41/ 722
1580	TCD Source Address (DMA_TCD44_SADDR)	32	R/W	0000_0000h	16.4.27/ 712
1584	TCD Transfer Attributes (DMA_TCD44_ATTR)	16	R/W	0000h	16.4.28/ 712
1586	TCD Signed Source Address Offset (DMA_TCD44_SOFF)	16	R/W	0000h	16.4.29/ 713
1588	TCD Minor Byte Count (Minor Loop Mapping Disabled) (DMA_TCD44_NBYTES_MLNO)	32	R/W	0000_0000h	16.4.30/ 713
1588	TCD Signed Minor Loop Offset (Minor Loop Mapping Enabled and Offset Disabled) (DMA_TCD44_NBYTES_MLOFFNO)	32	R/W	0000_0000h	16.4.31/ 714
1588	TCD Signed Minor Loop Offset (Minor Loop Mapping and Offset Enabled) (DMA_TCD44_NBYTES_MLOFFYES)	32	R/W	0000_0000h	16.4.32/ 715

Table continues on the next page...

DMA memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
158C	TCD Last Source Address Adjustment (DMA_TCD44_SLAST)	32	R/W	0000_0000h	16.4.33/ 716
1590	TCD Destination Address (DMA_TCD44_DADDR)	32	R/W	0000_0000h	16.4.34/ 717
1594	TCD Current Minor Loop Link, Major Loop Count (Channel Linking Enabled) (DMA_TCD44_CITER_ELINKYES)	16	R/W	0000h	16.4.35/ 717
1594	DMA_TCD44_CITER_ELINKNO	16	R/W	0000h	16.4.36/ 718
1596	TCD Signed Destination Address Offset (DMA_TCD44_DOFF)	16	R/W	0000h	16.4.37/ 719
1598	TCD Last Destination Address Adjustment/Scatter Gather Address (DMA_TCD44_DLASTSGA)	32	R/W	0000_0000h	16.4.38/ 720
159C	TCD Beginning Minor Loop Link, Major Loop Count (Channel Linking Enabled) (DMA_TCD44_BITER_ELINKYES)	16	R/W	0000h	16.4.39/ 720
159C	TCD Beginning Minor Loop Link, Major Loop Count (Channel Linking Disabled) (DMA_TCD44_BITER_ELINKNO)	16	R/W	0000h	16.4.40/ 721
159E	TCD Control and Status (DMA_TCD44_CSR)	16	R/W	0000h	16.4.41/ 722
15A0	TCD Source Address (DMA_TCD45_SADDR)	32	R/W	0000_0000h	16.4.27/ 712
15A4	TCD Transfer Attributes (DMA_TCD45_ATTR)	16	R/W	0000h	16.4.28/ 712
15A6	TCD Signed Source Address Offset (DMA_TCD45_SOFF)	16	R/W	0000h	16.4.29/ 713
15A8	TCD Minor Byte Count (Minor Loop Mapping Disabled) (DMA_TCD45_NBYTES_MLNO)	32	R/W	0000_0000h	16.4.30/ 713
15A8	TCD Signed Minor Loop Offset (Minor Loop Mapping Enabled and Offset Disabled) (DMA_TCD45_NBYTES_MLOFFNO)	32	R/W	0000_0000h	16.4.31/ 714
15A8	TCD Signed Minor Loop Offset (Minor Loop Mapping and Offset Enabled) (DMA_TCD45_NBYTES_MLOFFYES)	32	R/W	0000_0000h	16.4.32/ 715
15AC	TCD Last Source Address Adjustment (DMA_TCD45_SLAST)	32	R/W	0000_0000h	16.4.33/ 716
15B0	TCD Destination Address (DMA_TCD45_DADDR)	32	R/W	0000_0000h	16.4.34/ 717
15B4	TCD Current Minor Loop Link, Major Loop Count (Channel Linking Enabled) (DMA_TCD45_CITER_ELINKYES)	16	R/W	0000h	16.4.35/ 717
15B4	DMA_TCD45_CITER_ELINKNO	16	R/W	0000h	16.4.36/ 718
15B6	TCD Signed Destination Address Offset (DMA_TCD45_DOFF)	16	R/W	0000h	16.4.37/ 719
15B8	TCD Last Destination Address Adjustment/Scatter Gather Address (DMA_TCD45_DLASTSGA)	32	R/W	0000_0000h	16.4.38/ 720

Table continues on the next page...

DMA memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
15BC	TCD Beginning Minor Loop Link, Major Loop Count (Channel Linking Enabled) (DMA_TCD45_BITER_ELINKYES)	16	R/W	0000h	16.4.39/720
15BC	TCD Beginning Minor Loop Link, Major Loop Count (Channel Linking Disabled) (DMA_TCD45_BITER_ELINKNO)	16	R/W	0000h	16.4.40/721
15BE	TCD Control and Status (DMA_TCD45_CSR)	16	R/W	0000h	16.4.41/722
15C0	TCD Source Address (DMA_TCD46_SADDR)	32	R/W	0000_0000h	16.4.27/712
15C4	TCD Transfer Attributes (DMA_TCD46_ATTR)	16	R/W	0000h	16.4.28/712
15C6	TCD Signed Source Address Offset (DMA_TCD46_SOFF)	16	R/W	0000h	16.4.29/713
15C8	TCD Minor Byte Count (Minor Loop Mapping Disabled) (DMA_TCD46_NBYTES_MLNO)	32	R/W	0000_0000h	16.4.30/713
15C8	TCD Signed Minor Loop Offset (Minor Loop Mapping Enabled and Offset Disabled) (DMA_TCD46_NBYTES_MLOFFNO)	32	R/W	0000_0000h	16.4.31/714
15C8	TCD Signed Minor Loop Offset (Minor Loop Mapping and Offset Enabled) (DMA_TCD46_NBYTES_MLOFFYES)	32	R/W	0000_0000h	16.4.32/715
15CC	TCD Last Source Address Adjustment (DMA_TCD46_SLAST)	32	R/W	0000_0000h	16.4.33/716
15D0	TCD Destination Address (DMA_TCD46_DADDR)	32	R/W	0000_0000h	16.4.34/717
15D4	TCD Current Minor Loop Link, Major Loop Count (Channel Linking Enabled) (DMA_TCD46_CITER_ELINKYES)	16	R/W	0000h	16.4.35/717
15D4	DMA_TCD46_CITER_ELINKNO	16	R/W	0000h	16.4.36/718
15D6	TCD Signed Destination Address Offset (DMA_TCD46_DOFF)	16	R/W	0000h	16.4.37/719
15D8	TCD Last Destination Address Adjustment/Scatter Gather Address (DMA_TCD46_DLASTSGA)	32	R/W	0000_0000h	16.4.38/720
15DC	TCD Beginning Minor Loop Link, Major Loop Count (Channel Linking Enabled) (DMA_TCD46_BITER_ELINKYES)	16	R/W	0000h	16.4.39/720
15DC	TCD Beginning Minor Loop Link, Major Loop Count (Channel Linking Disabled) (DMA_TCD46_BITER_ELINKNO)	16	R/W	0000h	16.4.40/721
15DE	TCD Control and Status (DMA_TCD46_CSR)	16	R/W	0000h	16.4.41/722
15E0	TCD Source Address (DMA_TCD47_SADDR)	32	R/W	0000_0000h	16.4.27/712
15E4	TCD Transfer Attributes (DMA_TCD47_ATTR)	16	R/W	0000h	16.4.28/712

Table continues on the next page...

DMA memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
15E6	TCD Signed Source Address Offset (DMA_TCD47_SOFF)	16	R/W	0000h	16.4.29/ 713
15E8	TCD Minor Byte Count (Minor Loop Mapping Disabled) (DMA_TCD47_NBYTES_MLNO)	32	R/W	0000_0000h	16.4.30/ 713
15E8	TCD Signed Minor Loop Offset (Minor Loop Mapping Enabled and Offset Disabled) (DMA_TCD47_NBYTES_MLOFFNO)	32	R/W	0000_0000h	16.4.31/ 714
15E8	TCD Signed Minor Loop Offset (Minor Loop Mapping and Offset Enabled) (DMA_TCD47_NBYTES_MLOFFYES)	32	R/W	0000_0000h	16.4.32/ 715
15EC	TCD Last Source Address Adjustment (DMA_TCD47_SLAST)	32	R/W	0000_0000h	16.4.33/ 716
15F0	TCD Destination Address (DMA_TCD47_DADDR)	32	R/W	0000_0000h	16.4.34/ 717
15F4	TCD Current Minor Loop Link, Major Loop Count (Channel Linking Enabled) (DMA_TCD47_CITER_ELINKYES)	16	R/W	0000h	16.4.35/ 717
15F4	DMA_TCD47_CITER_ELINKNO	16	R/W	0000h	16.4.36/ 718
15F6	TCD Signed Destination Address Offset (DMA_TCD47_DOFF)	16	R/W	0000h	16.4.37/ 719
15F8	TCD Last Destination Address Adjustment/Scatter Gather Address (DMA_TCD47_DLASTSGA)	32	R/W	0000_0000h	16.4.38/ 720
15FC	TCD Beginning Minor Loop Link, Major Loop Count (Channel Linking Enabled) (DMA_TCD47_BITER_ELINKYES)	16	R/W	0000h	16.4.39/ 720
15FC	TCD Beginning Minor Loop Link, Major Loop Count (Channel Linking Disabled) (DMA_TCD47_BITER_ELINKNO)	16	R/W	0000h	16.4.40/ 721
15FE	TCD Control and Status (DMA_TCD47_CSR)	16	R/W	0000h	16.4.41/ 722
1600	TCD Source Address (DMA_TCD48_SADDR)	32	R/W	0000_0000h	16.4.27/ 712
1604	TCD Transfer Attributes (DMA_TCD48_ATTR)	16	R/W	0000h	16.4.28/ 712
1606	TCD Signed Source Address Offset (DMA_TCD48_SOFF)	16	R/W	0000h	16.4.29/ 713
1608	TCD Minor Byte Count (Minor Loop Mapping Disabled) (DMA_TCD48_NBYTES_MLNO)	32	R/W	0000_0000h	16.4.30/ 713
1608	TCD Signed Minor Loop Offset (Minor Loop Mapping Enabled and Offset Disabled) (DMA_TCD48_NBYTES_MLOFFNO)	32	R/W	0000_0000h	16.4.31/ 714
1608	TCD Signed Minor Loop Offset (Minor Loop Mapping and Offset Enabled) (DMA_TCD48_NBYTES_MLOFFYES)	32	R/W	0000_0000h	16.4.32/ 715
160C	TCD Last Source Address Adjustment (DMA_TCD48_SLAST)	32	R/W	0000_0000h	16.4.33/ 716
1610	TCD Destination Address (DMA_TCD48_DADDR)	32	R/W	0000_0000h	16.4.34/ 717

Table continues on the next page...

DMA memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
1614	TCD Current Minor Loop Link, Major Loop Count (Channel Linking Enabled) (DMA_TCD48_CITER_ELINKYES)	16	R/W	0000h	16.4.35/ 717
1614	DMA_TCD48_CITER_ELINKNO	16	R/W	0000h	16.4.36/ 718
1616	TCD Signed Destination Address Offset (DMA_TCD48_DOFF)	16	R/W	0000h	16.4.37/ 719
1618	TCD Last Destination Address Adjustment/Scatter Gather Address (DMA_TCD48_DLASTSGA)	32	R/W	0000_0000h	16.4.38/ 720
161C	TCD Beginning Minor Loop Link, Major Loop Count (Channel Linking Enabled) (DMA_TCD48_BITER_ELINKYES)	16	R/W	0000h	16.4.39/ 720
161C	TCD Beginning Minor Loop Link, Major Loop Count (Channel Linking Disabled) (DMA_TCD48_BITER_ELINKNO)	16	R/W	0000h	16.4.40/ 721
161E	TCD Control and Status (DMA_TCD48_CSR)	16	R/W	0000h	16.4.41/ 722
1620	TCD Source Address (DMA_TCD49_SADDR)	32	R/W	0000_0000h	16.4.27/ 712
1624	TCD Transfer Attributes (DMA_TCD49_ATTR)	16	R/W	0000h	16.4.28/ 712
1626	TCD Signed Source Address Offset (DMA_TCD49_SOFF)	16	R/W	0000h	16.4.29/ 713
1628	TCD Minor Byte Count (Minor Loop Mapping Disabled) (DMA_TCD49_NBYTES_MLNO)	32	R/W	0000_0000h	16.4.30/ 713
1628	TCD Signed Minor Loop Offset (Minor Loop Mapping Enabled and Offset Disabled) (DMA_TCD49_NBYTES_MLOFFNO)	32	R/W	0000_0000h	16.4.31/ 714
1628	TCD Signed Minor Loop Offset (Minor Loop Mapping and Offset Enabled) (DMA_TCD49_NBYTES_MLOFFYES)	32	R/W	0000_0000h	16.4.32/ 715
162C	TCD Last Source Address Adjustment (DMA_TCD49_SLAST)	32	R/W	0000_0000h	16.4.33/ 716
1630	TCD Destination Address (DMA_TCD49_DADDR)	32	R/W	0000_0000h	16.4.34/ 717
1634	TCD Current Minor Loop Link, Major Loop Count (Channel Linking Enabled) (DMA_TCD49_CITER_ELINKYES)	16	R/W	0000h	16.4.35/ 717
1634	DMA_TCD49_CITER_ELINKNO	16	R/W	0000h	16.4.36/ 718
1636	TCD Signed Destination Address Offset (DMA_TCD49_DOFF)	16	R/W	0000h	16.4.37/ 719
1638	TCD Last Destination Address Adjustment/Scatter Gather Address (DMA_TCD49_DLASTSGA)	32	R/W	0000_0000h	16.4.38/ 720
163C	TCD Beginning Minor Loop Link, Major Loop Count (Channel Linking Enabled) (DMA_TCD49_BITER_ELINKYES)	16	R/W	0000h	16.4.39/ 720

Table continues on the next page...

DMA memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
163C	TCD Beginning Minor Loop Link, Major Loop Count (Channel Linking Disabled) (DMA_TCD49_BITER_ELINKNO)	16	R/W	0000h	16.4.40/ 721
163E	TCD Control and Status (DMA_TCD49_CSR)	16	R/W	0000h	16.4.41/ 722
1640	TCD Source Address (DMA_TCD50_SADDR)	32	R/W	0000_0000h	16.4.27/ 712
1644	TCD Transfer Attributes (DMA_TCD50_ATTR)	16	R/W	0000h	16.4.28/ 712
1646	TCD Signed Source Address Offset (DMA_TCD50_SOFF)	16	R/W	0000h	16.4.29/ 713
1648	TCD Minor Byte Count (Minor Loop Mapping Disabled) (DMA_TCD50_NBYTES_MLNO)	32	R/W	0000_0000h	16.4.30/ 713
1648	TCD Signed Minor Loop Offset (Minor Loop Mapping Enabled and Offset Disabled) (DMA_TCD50_NBYTES_MLOFFNO)	32	R/W	0000_0000h	16.4.31/ 714
1648	TCD Signed Minor Loop Offset (Minor Loop Mapping and Offset Enabled) (DMA_TCD50_NBYTES_MLOFFYES)	32	R/W	0000_0000h	16.4.32/ 715
164C	TCD Last Source Address Adjustment (DMA_TCD50_SLAST)	32	R/W	0000_0000h	16.4.33/ 716
1650	TCD Destination Address (DMA_TCD50_DADDR)	32	R/W	0000_0000h	16.4.34/ 717
1654	TCD Current Minor Loop Link, Major Loop Count (Channel Linking Enabled) (DMA_TCD50_CITER_ELINKYES)	16	R/W	0000h	16.4.35/ 717
1654	DMA_TCD50_CITER_ELINKNO	16	R/W	0000h	16.4.36/ 718
1656	TCD Signed Destination Address Offset (DMA_TCD50_DOFF)	16	R/W	0000h	16.4.37/ 719
1658	TCD Last Destination Address Adjustment/Scatter Gather Address (DMA_TCD50_DLASTSGA)	32	R/W	0000_0000h	16.4.38/ 720
165C	TCD Beginning Minor Loop Link, Major Loop Count (Channel Linking Enabled) (DMA_TCD50_BITER_ELINKYES)	16	R/W	0000h	16.4.39/ 720
165C	TCD Beginning Minor Loop Link, Major Loop Count (Channel Linking Disabled) (DMA_TCD50_BITER_ELINKNO)	16	R/W	0000h	16.4.40/ 721
165E	TCD Control and Status (DMA_TCD50_CSR)	16	R/W	0000h	16.4.41/ 722
1660	TCD Source Address (DMA_TCD51_SADDR)	32	R/W	0000_0000h	16.4.27/ 712
1664	TCD Transfer Attributes (DMA_TCD51_ATTR)	16	R/W	0000h	16.4.28/ 712
1666	TCD Signed Source Address Offset (DMA_TCD51_SOFF)	16	R/W	0000h	16.4.29/ 713
1668	TCD Minor Byte Count (Minor Loop Mapping Disabled) (DMA_TCD51_NBYTES_MLNO)	32	R/W	0000_0000h	16.4.30/ 713

Table continues on the next page...

DMA memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
1668	TCD Signed Minor Loop Offset (Minor Loop Mapping Enabled and Offset Disabled) (DMA_TCD51_NBYTES_MLOFFNO)	32	R/W	0000_0000h	16.4.31/ 714
1668	TCD Signed Minor Loop Offset (Minor Loop Mapping and Offset Enabled) (DMA_TCD51_NBYTES_MLOFFYES)	32	R/W	0000_0000h	16.4.32/ 715
166C	TCD Last Source Address Adjustment (DMA_TCD51_SLAST)	32	R/W	0000_0000h	16.4.33/ 716
1670	TCD Destination Address (DMA_TCD51_DADDR)	32	R/W	0000_0000h	16.4.34/ 717
1674	TCD Current Minor Loop Link, Major Loop Count (Channel Linking Enabled) (DMA_TCD51_CITER_ELINKYES)	16	R/W	0000h	16.4.35/ 717
1674	DMA_TCD51_CITER_ELINKNO	16	R/W	0000h	16.4.36/ 718
1676	TCD Signed Destination Address Offset (DMA_TCD51_DOFF)	16	R/W	0000h	16.4.37/ 719
1678	TCD Last Destination Address Adjustment/Scatter Gather Address (DMA_TCD51_DLASTSGA)	32	R/W	0000_0000h	16.4.38/ 720
167C	TCD Beginning Minor Loop Link, Major Loop Count (Channel Linking Enabled) (DMA_TCD51_BITER_ELINKYES)	16	R/W	0000h	16.4.39/ 720
167C	TCD Beginning Minor Loop Link, Major Loop Count (Channel Linking Disabled) (DMA_TCD51_BITER_ELINKNO)	16	R/W	0000h	16.4.40/ 721
167E	TCD Control and Status (DMA_TCD51_CSR)	16	R/W	0000h	16.4.41/ 722
1680	TCD Source Address (DMA_TCD52_SADDR)	32	R/W	0000_0000h	16.4.27/ 712
1684	TCD Transfer Attributes (DMA_TCD52_ATTR)	16	R/W	0000h	16.4.28/ 712
1686	TCD Signed Source Address Offset (DMA_TCD52_SOFF)	16	R/W	0000h	16.4.29/ 713
1688	TCD Minor Byte Count (Minor Loop Mapping Disabled) (DMA_TCD52_NBYTES_MLNO)	32	R/W	0000_0000h	16.4.30/ 713
1688	TCD Signed Minor Loop Offset (Minor Loop Mapping Enabled and Offset Disabled) (DMA_TCD52_NBYTES_MLOFFNO)	32	R/W	0000_0000h	16.4.31/ 714
1688	TCD Signed Minor Loop Offset (Minor Loop Mapping and Offset Enabled) (DMA_TCD52_NBYTES_MLOFFYES)	32	R/W	0000_0000h	16.4.32/ 715
168C	TCD Last Source Address Adjustment (DMA_TCD52_SLAST)	32	R/W	0000_0000h	16.4.33/ 716
1690	TCD Destination Address (DMA_TCD52_DADDR)	32	R/W	0000_0000h	16.4.34/ 717
1694	TCD Current Minor Loop Link, Major Loop Count (Channel Linking Enabled) (DMA_TCD52_CITER_ELINKYES)	16	R/W	0000h	16.4.35/ 717
1694	DMA_TCD52_CITER_ELINKNO	16	R/W	0000h	16.4.36/ 718

Table continues on the next page...

DMA memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
1696	TCD Signed Destination Address Offset (DMA_TCD52_DOFF)	16	R/W	0000h	16.4.37/ 719
1698	TCD Last Destination Address Adjustment/Scatter Gather Address (DMA_TCD52_DLASTSGA)	32	R/W	0000_0000h	16.4.38/ 720
169C	TCD Beginning Minor Loop Link, Major Loop Count (Channel Linking Enabled) (DMA_TCD52_BITER_ELINKYES)	16	R/W	0000h	16.4.39/ 720
169C	TCD Beginning Minor Loop Link, Major Loop Count (Channel Linking Disabled) (DMA_TCD52_BITER_ELINKNO)	16	R/W	0000h	16.4.40/ 721
169E	TCD Control and Status (DMA_TCD52_CSR)	16	R/W	0000h	16.4.41/ 722
16A0	TCD Source Address (DMA_TCD53_SADDR)	32	R/W	0000_0000h	16.4.27/ 712
16A4	TCD Transfer Attributes (DMA_TCD53_ATTR)	16	R/W	0000h	16.4.28/ 712
16A6	TCD Signed Source Address Offset (DMA_TCD53_SOFF)	16	R/W	0000h	16.4.29/ 713
16A8	TCD Minor Byte Count (Minor Loop Mapping Disabled) (DMA_TCD53_NBYTES_MLNO)	32	R/W	0000_0000h	16.4.30/ 713
16A8	TCD Signed Minor Loop Offset (Minor Loop Mapping Enabled and Offset Disabled) (DMA_TCD53_NBYTES_MLOFFNO)	32	R/W	0000_0000h	16.4.31/ 714
16A8	TCD Signed Minor Loop Offset (Minor Loop Mapping and Offset Enabled) (DMA_TCD53_NBYTES_MLOFFYES)	32	R/W	0000_0000h	16.4.32/ 715
16AC	TCD Last Source Address Adjustment (DMA_TCD53_SLAST)	32	R/W	0000_0000h	16.4.33/ 716
16B0	TCD Destination Address (DMA_TCD53_DADDR)	32	R/W	0000_0000h	16.4.34/ 717
16B4	TCD Current Minor Loop Link, Major Loop Count (Channel Linking Enabled) (DMA_TCD53_CITER_ELINKYES)	16	R/W	0000h	16.4.35/ 717
16B4	DMA_TCD53_CITER_ELINKNO	16	R/W	0000h	16.4.36/ 718
16B6	TCD Signed Destination Address Offset (DMA_TCD53_DOFF)	16	R/W	0000h	16.4.37/ 719
16B8	TCD Last Destination Address Adjustment/Scatter Gather Address (DMA_TCD53_DLASTSGA)	32	R/W	0000_0000h	16.4.38/ 720
16BC	TCD Beginning Minor Loop Link, Major Loop Count (Channel Linking Enabled) (DMA_TCD53_BITER_ELINKYES)	16	R/W	0000h	16.4.39/ 720
16BC	TCD Beginning Minor Loop Link, Major Loop Count (Channel Linking Disabled) (DMA_TCD53_BITER_ELINKNO)	16	R/W	0000h	16.4.40/ 721
16BE	TCD Control and Status (DMA_TCD53_CSR)	16	R/W	0000h	16.4.41/ 722

Table continues on the next page...

DMA memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
16C0	TCD Source Address (DMA_TCD54_SADDR)	32	R/W	0000_0000h	16.4.27/ 712
16C4	TCD Transfer Attributes (DMA_TCD54_ATTR)	16	R/W	0000h	16.4.28/ 712
16C6	TCD Signed Source Address Offset (DMA_TCD54_SOFF)	16	R/W	0000h	16.4.29/ 713
16C8	TCD Minor Byte Count (Minor Loop Mapping Disabled) (DMA_TCD54_NBYTES_MLNO)	32	R/W	0000_0000h	16.4.30/ 713
16C8	TCD Signed Minor Loop Offset (Minor Loop Mapping Enabled and Offset Disabled) (DMA_TCD54_NBYTES_MLOFFNO)	32	R/W	0000_0000h	16.4.31/ 714
16C8	TCD Signed Minor Loop Offset (Minor Loop Mapping and Offset Enabled) (DMA_TCD54_NBYTES_MLOFFYES)	32	R/W	0000_0000h	16.4.32/ 715
16CC	TCD Last Source Address Adjustment (DMA_TCD54_SLAST)	32	R/W	0000_0000h	16.4.33/ 716
16D0	TCD Destination Address (DMA_TCD54_DADDR)	32	R/W	0000_0000h	16.4.34/ 717
16D4	TCD Current Minor Loop Link, Major Loop Count (Channel Linking Enabled) (DMA_TCD54_CITER_ELINKYES)	16	R/W	0000h	16.4.35/ 717
16D4	DMA_TCD54_CITER_ELINKNO	16	R/W	0000h	16.4.36/ 718
16D6	TCD Signed Destination Address Offset (DMA_TCD54_DOFF)	16	R/W	0000h	16.4.37/ 719
16D8	TCD Last Destination Address Adjustment/Scatter Gather Address (DMA_TCD54_DLASTSGA)	32	R/W	0000_0000h	16.4.38/ 720
16DC	TCD Beginning Minor Loop Link, Major Loop Count (Channel Linking Enabled) (DMA_TCD54_BITER_ELINKYES)	16	R/W	0000h	16.4.39/ 720
16DC	TCD Beginning Minor Loop Link, Major Loop Count (Channel Linking Disabled) (DMA_TCD54_BITER_ELINKNO)	16	R/W	0000h	16.4.40/ 721
16DE	TCD Control and Status (DMA_TCD54_CSR)	16	R/W	0000h	16.4.41/ 722
16E0	TCD Source Address (DMA_TCD55_SADDR)	32	R/W	0000_0000h	16.4.27/ 712
16E4	TCD Transfer Attributes (DMA_TCD55_ATTR)	16	R/W	0000h	16.4.28/ 712
16E6	TCD Signed Source Address Offset (DMA_TCD55_SOFF)	16	R/W	0000h	16.4.29/ 713
16E8	TCD Minor Byte Count (Minor Loop Mapping Disabled) (DMA_TCD55_NBYTES_MLNO)	32	R/W	0000_0000h	16.4.30/ 713
16E8	TCD Signed Minor Loop Offset (Minor Loop Mapping Enabled and Offset Disabled) (DMA_TCD55_NBYTES_MLOFFNO)	32	R/W	0000_0000h	16.4.31/ 714
16E8	TCD Signed Minor Loop Offset (Minor Loop Mapping and Offset Enabled) (DMA_TCD55_NBYTES_MLOFFYES)	32	R/W	0000_0000h	16.4.32/ 715

Table continues on the next page...

DMA memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
16EC	TCD Last Source Address Adjustment (DMA_TCD55_SLAST)	32	R/W	0000_0000h	16.4.33/ 716
16F0	TCD Destination Address (DMA_TCD55_DADDR)	32	R/W	0000_0000h	16.4.34/ 717
16F4	TCD Current Minor Loop Link, Major Loop Count (Channel Linking Enabled) (DMA_TCD55_CITER_ELINKYES)	16	R/W	0000h	16.4.35/ 717
16F4	DMA_TCD55_CITER_ELINKNO	16	R/W	0000h	16.4.36/ 718
16F6	TCD Signed Destination Address Offset (DMA_TCD55_DOFF)	16	R/W	0000h	16.4.37/ 719
16F8	TCD Last Destination Address Adjustment/Scatter Gather Address (DMA_TCD55_DLASTGA)	32	R/W	0000_0000h	16.4.38/ 720
16FC	TCD Beginning Minor Loop Link, Major Loop Count (Channel Linking Enabled) (DMA_TCD55_BITER_ELINKYES)	16	R/W	0000h	16.4.39/ 720
16FC	TCD Beginning Minor Loop Link, Major Loop Count (Channel Linking Disabled) (DMA_TCD55_BITER_ELINKNO)	16	R/W	0000h	16.4.40/ 721
16FE	TCD Control and Status (DMA_TCD55_CSR)	16	R/W	0000h	16.4.41/ 722
1700	TCD Source Address (DMA_TCD56_SADDR)	32	R/W	0000_0000h	16.4.27/ 712
1704	TCD Transfer Attributes (DMA_TCD56_ATTR)	16	R/W	0000h	16.4.28/ 712
1706	TCD Signed Source Address Offset (DMA_TCD56_SOFF)	16	R/W	0000h	16.4.29/ 713
1708	TCD Minor Byte Count (Minor Loop Mapping Disabled) (DMA_TCD56_NBYTES_MLNO)	32	R/W	0000_0000h	16.4.30/ 713
1708	TCD Signed Minor Loop Offset (Minor Loop Mapping Enabled and Offset Disabled) (DMA_TCD56_NBYTES_MLOFFNO)	32	R/W	0000_0000h	16.4.31/ 714
1708	TCD Signed Minor Loop Offset (Minor Loop Mapping and Offset Enabled) (DMA_TCD56_NBYTES_MLOFFYES)	32	R/W	0000_0000h	16.4.32/ 715
170C	TCD Last Source Address Adjustment (DMA_TCD56_SLAST)	32	R/W	0000_0000h	16.4.33/ 716
1710	TCD Destination Address (DMA_TCD56_DADDR)	32	R/W	0000_0000h	16.4.34/ 717
1714	TCD Current Minor Loop Link, Major Loop Count (Channel Linking Enabled) (DMA_TCD56_CITER_ELINKYES)	16	R/W	0000h	16.4.35/ 717
1714	DMA_TCD56_CITER_ELINKNO	16	R/W	0000h	16.4.36/ 718
1716	TCD Signed Destination Address Offset (DMA_TCD56_DOFF)	16	R/W	0000h	16.4.37/ 719
1718	TCD Last Destination Address Adjustment/Scatter Gather Address (DMA_TCD56_DLASTGA)	32	R/W	0000_0000h	16.4.38/ 720

Table continues on the next page...

DMA memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
171C	TCD Beginning Minor Loop Link, Major Loop Count (Channel Linking Enabled) (DMA_TCD56_BITER_ELINKYES)	16	R/W	0000h	16.4.39/ 720
171C	TCD Beginning Minor Loop Link, Major Loop Count (Channel Linking Disabled) (DMA_TCD56_BITER_ELINKNO)	16	R/W	0000h	16.4.40/ 721
171E	TCD Control and Status (DMA_TCD56_CSR)	16	R/W	0000h	16.4.41/ 722
1720	TCD Source Address (DMA_TCD57_SADDR)	32	R/W	0000_0000h	16.4.27/ 712
1724	TCD Transfer Attributes (DMA_TCD57_ATTR)	16	R/W	0000h	16.4.28/ 712
1726	TCD Signed Source Address Offset (DMA_TCD57_SOFF)	16	R/W	0000h	16.4.29/ 713
1728	TCD Minor Byte Count (Minor Loop Mapping Disabled) (DMA_TCD57_NBYTES_MLNO)	32	R/W	0000_0000h	16.4.30/ 713
1728	TCD Signed Minor Loop Offset (Minor Loop Mapping Enabled and Offset Disabled) (DMA_TCD57_NBYTES_MLOFFNO)	32	R/W	0000_0000h	16.4.31/ 714
1728	TCD Signed Minor Loop Offset (Minor Loop Mapping and Offset Enabled) (DMA_TCD57_NBYTES_MLOFFYES)	32	R/W	0000_0000h	16.4.32/ 715
172C	TCD Last Source Address Adjustment (DMA_TCD57_SLAST)	32	R/W	0000_0000h	16.4.33/ 716
1730	TCD Destination Address (DMA_TCD57_DADDR)	32	R/W	0000_0000h	16.4.34/ 717
1734	TCD Current Minor Loop Link, Major Loop Count (Channel Linking Enabled) (DMA_TCD57_CITER_ELINKYES)	16	R/W	0000h	16.4.35/ 717
1734	DMA_TCD57_CITER_ELINKNO	16	R/W	0000h	16.4.36/ 718
1736	TCD Signed Destination Address Offset (DMA_TCD57_DOFF)	16	R/W	0000h	16.4.37/ 719
1738	TCD Last Destination Address Adjustment/Scatter Gather Address (DMA_TCD57_DLASTSGA)	32	R/W	0000_0000h	16.4.38/ 720
173C	TCD Beginning Minor Loop Link, Major Loop Count (Channel Linking Enabled) (DMA_TCD57_BITER_ELINKYES)	16	R/W	0000h	16.4.39/ 720
173C	TCD Beginning Minor Loop Link, Major Loop Count (Channel Linking Disabled) (DMA_TCD57_BITER_ELINKNO)	16	R/W	0000h	16.4.40/ 721
173E	TCD Control and Status (DMA_TCD57_CSR)	16	R/W	0000h	16.4.41/ 722
1740	TCD Source Address (DMA_TCD58_SADDR)	32	R/W	0000_0000h	16.4.27/ 712
1744	TCD Transfer Attributes (DMA_TCD58_ATTR)	16	R/W	0000h	16.4.28/ 712

Table continues on the next page...

DMA memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
1746	TCD Signed Source Address Offset (DMA_TCD58_SOFF)	16	R/W	0000h	16.4.29/ 713
1748	TCD Minor Byte Count (Minor Loop Mapping Disabled) (DMA_TCD58_NBYTES_MLNO)	32	R/W	0000_0000h	16.4.30/ 713
1748	TCD Signed Minor Loop Offset (Minor Loop Mapping Enabled and Offset Disabled) (DMA_TCD58_NBYTES_MLOFFNO)	32	R/W	0000_0000h	16.4.31/ 714
1748	TCD Signed Minor Loop Offset (Minor Loop Mapping and Offset Enabled) (DMA_TCD58_NBYTES_MLOFFYES)	32	R/W	0000_0000h	16.4.32/ 715
174C	TCD Last Source Address Adjustment (DMA_TCD58_SLAST)	32	R/W	0000_0000h	16.4.33/ 716
1750	TCD Destination Address (DMA_TCD58_DADDR)	32	R/W	0000_0000h	16.4.34/ 717
1754	TCD Current Minor Loop Link, Major Loop Count (Channel Linking Enabled) (DMA_TCD58_CITER_ELINKYES)	16	R/W	0000h	16.4.35/ 717
1754	DMA_TCD58_CITER_ELINKNO	16	R/W	0000h	16.4.36/ 718
1756	TCD Signed Destination Address Offset (DMA_TCD58_DOFF)	16	R/W	0000h	16.4.37/ 719
1758	TCD Last Destination Address Adjustment/Scatter Gather Address (DMA_TCD58_DLASTSGA)	32	R/W	0000_0000h	16.4.38/ 720
175C	TCD Beginning Minor Loop Link, Major Loop Count (Channel Linking Enabled) (DMA_TCD58_BITER_ELINKYES)	16	R/W	0000h	16.4.39/ 720
175C	TCD Beginning Minor Loop Link, Major Loop Count (Channel Linking Disabled) (DMA_TCD58_BITER_ELINKNO)	16	R/W	0000h	16.4.40/ 721
175E	TCD Control and Status (DMA_TCD58_CSR)	16	R/W	0000h	16.4.41/ 722
1760	TCD Source Address (DMA_TCD59_SADDR)	32	R/W	0000_0000h	16.4.27/ 712
1764	TCD Transfer Attributes (DMA_TCD59_ATTR)	16	R/W	0000h	16.4.28/ 712
1766	TCD Signed Source Address Offset (DMA_TCD59_SOFF)	16	R/W	0000h	16.4.29/ 713
1768	TCD Minor Byte Count (Minor Loop Mapping Disabled) (DMA_TCD59_NBYTES_MLNO)	32	R/W	0000_0000h	16.4.30/ 713
1768	TCD Signed Minor Loop Offset (Minor Loop Mapping Enabled and Offset Disabled) (DMA_TCD59_NBYTES_MLOFFNO)	32	R/W	0000_0000h	16.4.31/ 714
1768	TCD Signed Minor Loop Offset (Minor Loop Mapping and Offset Enabled) (DMA_TCD59_NBYTES_MLOFFYES)	32	R/W	0000_0000h	16.4.32/ 715
176C	TCD Last Source Address Adjustment (DMA_TCD59_SLAST)	32	R/W	0000_0000h	16.4.33/ 716
1770	TCD Destination Address (DMA_TCD59_DADDR)	32	R/W	0000_0000h	16.4.34/ 717

Table continues on the next page...

DMA memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
1774	TCD Current Minor Loop Link, Major Loop Count (Channel Linking Enabled) (DMA_TCD59_CITER_ELINKYES)	16	R/W	0000h	16.4.35/ 717
1774	DMA_TCD59_CITER_ELINKNO	16	R/W	0000h	16.4.36/ 718
1776	TCD Signed Destination Address Offset (DMA_TCD59_DOFF)	16	R/W	0000h	16.4.37/ 719
1778	TCD Last Destination Address Adjustment/Scatter Gather Address (DMA_TCD59_DLASTSGA)	32	R/W	0000_0000h	16.4.38/ 720
177C	TCD Beginning Minor Loop Link, Major Loop Count (Channel Linking Enabled) (DMA_TCD59_BITER_ELINKYES)	16	R/W	0000h	16.4.39/ 720
177C	TCD Beginning Minor Loop Link, Major Loop Count (Channel Linking Disabled) (DMA_TCD59_BITER_ELINKNO)	16	R/W	0000h	16.4.40/ 721
177E	TCD Control and Status (DMA_TCD59_CSR)	16	R/W	0000h	16.4.41/ 722
1780	TCD Source Address (DMA_TCD60_SADDR)	32	R/W	0000_0000h	16.4.27/ 712
1784	TCD Transfer Attributes (DMA_TCD60_ATTR)	16	R/W	0000h	16.4.28/ 712
1786	TCD Signed Source Address Offset (DMA_TCD60_SOFF)	16	R/W	0000h	16.4.29/ 713
1788	TCD Minor Byte Count (Minor Loop Mapping Disabled) (DMA_TCD60_NBYTES_MLNO)	32	R/W	0000_0000h	16.4.30/ 713
1788	TCD Signed Minor Loop Offset (Minor Loop Mapping Enabled and Offset Disabled) (DMA_TCD60_NBYTES_MLOFFNO)	32	R/W	0000_0000h	16.4.31/ 714
1788	TCD Signed Minor Loop Offset (Minor Loop Mapping and Offset Enabled) (DMA_TCD60_NBYTES_MLOFFYES)	32	R/W	0000_0000h	16.4.32/ 715
178C	TCD Last Source Address Adjustment (DMA_TCD60_SLAST)	32	R/W	0000_0000h	16.4.33/ 716
1790	TCD Destination Address (DMA_TCD60_DADDR)	32	R/W	0000_0000h	16.4.34/ 717
1794	TCD Current Minor Loop Link, Major Loop Count (Channel Linking Enabled) (DMA_TCD60_CITER_ELINKYES)	16	R/W	0000h	16.4.35/ 717
1794	DMA_TCD60_CITER_ELINKNO	16	R/W	0000h	16.4.36/ 718
1796	TCD Signed Destination Address Offset (DMA_TCD60_DOFF)	16	R/W	0000h	16.4.37/ 719
1798	TCD Last Destination Address Adjustment/Scatter Gather Address (DMA_TCD60_DLASTSGA)	32	R/W	0000_0000h	16.4.38/ 720
179C	TCD Beginning Minor Loop Link, Major Loop Count (Channel Linking Enabled) (DMA_TCD60_BITER_ELINKYES)	16	R/W	0000h	16.4.39/ 720

Table continues on the next page...

DMA memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
179C	TCD Beginning Minor Loop Link, Major Loop Count (Channel Linking Disabled) (DMA_TCD60_BITER_ELINKNO)	16	R/W	0000h	16.4.40/ 721
179E	TCD Control and Status (DMA_TCD60_CSR)	16	R/W	0000h	16.4.41/ 722
17A0	TCD Source Address (DMA_TCD61_SADDR)	32	R/W	0000_0000h	16.4.27/ 712
17A4	TCD Transfer Attributes (DMA_TCD61_ATTR)	16	R/W	0000h	16.4.28/ 712
17A6	TCD Signed Source Address Offset (DMA_TCD61_SOFF)	16	R/W	0000h	16.4.29/ 713
17A8	TCD Minor Byte Count (Minor Loop Mapping Disabled) (DMA_TCD61_NBYTES_MLNO)	32	R/W	0000_0000h	16.4.30/ 713
17A8	TCD Signed Minor Loop Offset (Minor Loop Mapping Enabled and Offset Disabled) (DMA_TCD61_NBYTES_MLOFFNO)	32	R/W	0000_0000h	16.4.31/ 714
17A8	TCD Signed Minor Loop Offset (Minor Loop Mapping and Offset Enabled) (DMA_TCD61_NBYTES_MLOFFYES)	32	R/W	0000_0000h	16.4.32/ 715
17AC	TCD Last Source Address Adjustment (DMA_TCD61_SLAST)	32	R/W	0000_0000h	16.4.33/ 716
17B0	TCD Destination Address (DMA_TCD61_DADDR)	32	R/W	0000_0000h	16.4.34/ 717
17B4	TCD Current Minor Loop Link, Major Loop Count (Channel Linking Enabled) (DMA_TCD61_CITER_ELINKYES)	16	R/W	0000h	16.4.35/ 717
17B4	DMA_TCD61_CITER_ELINKNO	16	R/W	0000h	16.4.36/ 718
17B6	TCD Signed Destination Address Offset (DMA_TCD61_DOFF)	16	R/W	0000h	16.4.37/ 719
17B8	TCD Last Destination Address Adjustment/Scatter Gather Address (DMA_TCD61_DLASTSGA)	32	R/W	0000_0000h	16.4.38/ 720
17BC	TCD Beginning Minor Loop Link, Major Loop Count (Channel Linking Enabled) (DMA_TCD61_BITER_ELINKYES)	16	R/W	0000h	16.4.39/ 720
17BC	TCD Beginning Minor Loop Link, Major Loop Count (Channel Linking Disabled) (DMA_TCD61_BITER_ELINKNO)	16	R/W	0000h	16.4.40/ 721
17BE	TCD Control and Status (DMA_TCD61_CSR)	16	R/W	0000h	16.4.41/ 722
17C0	TCD Source Address (DMA_TCD62_SADDR)	32	R/W	0000_0000h	16.4.27/ 712
17C4	TCD Transfer Attributes (DMA_TCD62_ATTR)	16	R/W	0000h	16.4.28/ 712
17C6	TCD Signed Source Address Offset (DMA_TCD62_SOFF)	16	R/W	0000h	16.4.29/ 713
17C8	TCD Minor Byte Count (Minor Loop Mapping Disabled) (DMA_TCD62_NBYTES_MLNO)	32	R/W	0000_0000h	16.4.30/ 713

Table continues on the next page...

DMA memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
17C8	TCD Signed Minor Loop Offset (Minor Loop Mapping Enabled and Offset Disabled) (DMA_TCD62_NBYTES_MLOFFNO)	32	R/W	0000_0000h	16.4.31/ 714
17C8	TCD Signed Minor Loop Offset (Minor Loop Mapping and Offset Enabled) (DMA_TCD62_NBYTES_MLOFFYES)	32	R/W	0000_0000h	16.4.32/ 715
17CC	TCD Last Source Address Adjustment (DMA_TCD62_SLAST)	32	R/W	0000_0000h	16.4.33/ 716
17D0	TCD Destination Address (DMA_TCD62_DADDR)	32	R/W	0000_0000h	16.4.34/ 717
17D4	TCD Current Minor Loop Link, Major Loop Count (Channel Linking Enabled) (DMA_TCD62_CITER_ELINKYES)	16	R/W	0000h	16.4.35/ 717
17D4	DMA_TCD62_CITER_ELINKNO	16	R/W	0000h	16.4.36/ 718
17D6	TCD Signed Destination Address Offset (DMA_TCD62_DOFF)	16	R/W	0000h	16.4.37/ 719
17D8	TCD Last Destination Address Adjustment/Scatter Gather Address (DMA_TCD62_DLASTSGA)	32	R/W	0000_0000h	16.4.38/ 720
17DC	TCD Beginning Minor Loop Link, Major Loop Count (Channel Linking Enabled) (DMA_TCD62_BITER_ELINKYES)	16	R/W	0000h	16.4.39/ 720
17DC	TCD Beginning Minor Loop Link, Major Loop Count (Channel Linking Disabled) (DMA_TCD62_BITER_ELINKNO)	16	R/W	0000h	16.4.40/ 721
17DE	TCD Control and Status (DMA_TCD62_CSR)	16	R/W	0000h	16.4.41/ 722
17E0	TCD Source Address (DMA_TCD63_SADDR)	32	R/W	0000_0000h	16.4.27/ 712
17E4	TCD Transfer Attributes (DMA_TCD63_ATTR)	16	R/W	0000h	16.4.28/ 712
17E6	TCD Signed Source Address Offset (DMA_TCD63_SOFF)	16	R/W	0000h	16.4.29/ 713
17E8	TCD Minor Byte Count (Minor Loop Mapping Disabled) (DMA_TCD63_NBYTES_MLNO)	32	R/W	0000_0000h	16.4.30/ 713
17E8	TCD Signed Minor Loop Offset (Minor Loop Mapping Enabled and Offset Disabled) (DMA_TCD63_NBYTES_MLOFFNO)	32	R/W	0000_0000h	16.4.31/ 714
17E8	TCD Signed Minor Loop Offset (Minor Loop Mapping and Offset Enabled) (DMA_TCD63_NBYTES_MLOFFYES)	32	R/W	0000_0000h	16.4.32/ 715
17EC	TCD Last Source Address Adjustment (DMA_TCD63_SLAST)	32	R/W	0000_0000h	16.4.33/ 716
17F0	TCD Destination Address (DMA_TCD63_DADDR)	32	R/W	0000_0000h	16.4.34/ 717
17F4	TCD Current Minor Loop Link, Major Loop Count (Channel Linking Enabled) (DMA_TCD63_CITER_ELINKYES)	16	R/W	0000h	16.4.35/ 717
17F4	DMA_TCD63_CITER_ELINKNO	16	R/W	0000h	16.4.36/ 718

Table continues on the next page...

DMA memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
17F6	TCD Signed Destination Address Offset (DMA_TCD63_DOFF)	16	R/W	0000h	16.4.37/719
17F8	TCD Last Destination Address Adjustment/Scatter Gather Address (DMA_TCD63_DLASTSGA)	32	R/W	0000_0000h	16.4.38/720
17FC	TCD Beginning Minor Loop Link, Major Loop Count (Channel Linking Enabled) (DMA_TCD63_BITER_ELINKYES)	16	R/W	0000h	16.4.39/720
17FC	TCD Beginning Minor Loop Link, Major Loop Count (Channel Linking Disabled) (DMA_TCD63_BITER_ELINKNO)	16	R/W	0000h	16.4.40/721
17FE	TCD Control and Status (DMA_TCD63_CSR)	16	R/W	0000h	16.4.41/722

16.4.5 Control Register (DMA_CR)

The CR defines the basic operating configuration of the DMA. The DMA arbitrates channel service requests in four groups of 16 channels each:

- Group 3 contains channels 63-48
- Group 2 contains channels 47-32
- Group 1 contains channels 31-16
- Group 0 contains channels 15-0

Arbitration within a group can be configured to use either a fixed-priority or a round-robin scheme. For fixed-priority arbitration, the highest priority channel requesting service is selected to execute. The channel priority registers assign the priorities; see the DCHPRIn registers. For round-robin arbitration, the channel priorities are ignored and channels within each group are cycled through (from high to low channel number) without regard to priority.

NOTE

For correct operation, writes to the CR register must be performed only when the DMA channels are inactive; that is, when TCDn_CSR[ACTIVE] bits are cleared.

The group priorities operate in a similar fashion. In group fixed priority arbitration mode, channel service requests in the highest priority group are executed first, where priority level 3 is the highest and priority level 0 is the lowest. The group priorities are assigned in the GRPnPRI fields of the DMA Control Register (CR). All group priorities must have unique values prior to any channel service requests occurring; otherwise, a configuration

error will be reported. For group round robin arbitration, the group priorities are ignored and the groups are cycled through (from high to low group number) without regard to priority.

Minor loop offsets are address offset values added to the final source address (TCDn_SADDR) or destination address (TCDn_DADDR) upon minor loop completion. When minor loop offsets are enabled, the minor loop offset (MLOFF) is added to the final source address (TCDn_SADDR), to the final destination address (TCDn_DADDR), or to both prior to the addresses being written back into the TCD. If the major loop is complete, the minor loop offset is ignored and the major loop address offsets (TCDn_SLAST and TCDn_DLAST_SGA) are used to compute the next TCDn_SADDR and TCDn_DADDR values.

When minor loop mapping is enabled (EMLM is 1), TCDn word2 is redefined. A portion of TCDn word2 is used to specify multiple fields: a source enable bit (SMLOE) to specify the minor loop offset should be applied to the source address (TCDn_SADDR) upon minor loop completion, a destination enable bit (DMLOE) to specify the minor loop offset should be applied to the destination address (TCDn_DADDR) upon minor loop completion, and the sign extended minor loop offset value (MLOFF). The same offset value (MLOFF) is used for both source and destination minor loop offsets. When either minor loop offset is enabled (SMLOE set or DMLOE set), the NBYTES field is reduced to 10 bits. When both minor loop offsets are disabled (SMLOE cleared and DMLOE cleared), the NBYTES field is a 30-bit vector.

When minor loop mapping is disabled (EMLM is 0), all 32 bits of TCDn word2 are assigned to the NBYTES field.

Address: 0h base + 0h offset = 0h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0													CX	ECX	
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	GRP3PRI		GRP2PRI		GRP1PRI		GRP0PRI		EMLM	CLM	HALT	HOE	ERGA	ERCA	EDBG	Reserved
W																
Reset	1	1	1	0	0	1	0	0	0	0	0	0	0	0	0	0

DMA_CR field descriptions

Field	Description
0–13 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
14 CX	Cancel Transfer 0 Normal operation 1 Cancel the remaining data transfer. Stop the executing channel and force the minor loop to finish. The cancel takes effect after the last write of the current read/write sequence. The CX bit clears itself after the cancel has been honored. This cancel retires the channel normally as if the minor loop was completed.
15 ECX	Error Cancel Transfer 0 Normal operation 1 Cancel the remaining data transfer in the same fashion as the CX bit. Stop the executing channel and force the minor loop to finish. The cancel takes effect after the last write of the current read/write sequence. The ECX bit clears itself after the cancel is honored. In addition to cancelling the transfer, ECX treats the cancel as an error condition, thus updating the Error Status register (DMAx_ES) and generating an optional error interrupt.
16–17 GRP3PRI	Channel Group 3 Priority Group 3 priority level when fixed priority group arbitration is enabled.
18–19 GRP2PRI	Channel Group 2 Priority Group 2 priority level when fixed priority group arbitration is enabled.
20–21 GRP1PRI	Channel Group 1 Priority Group 1 priority level when fixed priority group arbitration is enabled.
22–23 GRP0PRI	Channel Group 0 Priority Group 0 priority level when fixed priority group arbitration is enabled.
24 EMLM	Enable Minor Loop Mapping 0 Disabled. TCDn.word2 is defined as a 32-bit NBYTES field. 1 Enabled. TCDn.word2 is redefined to include individual enable fields, an offset field, and the NBYTES field. The individual enable fields allow the minor loop offset to be applied to the source address, the destination address, or both. The NBYTES field is reduced when either offset is enabled.
25 CLM	Continuous Link Mode NOTE: Do not use continuous link mode with a channel linking to itself if there is only one minor loop iteration per service request, e.g., if the channel's NBYTES value is the same as either the source or destination size. The same data transfer profile can be achieved by simply increasing the NBYTES value, which provides more efficient, faster processing. 0 A minor loop channel link made to itself goes through channel arbitration before being activated again. 1 A minor loop channel link made to itself does not go through channel arbitration before being activated again. Upon minor loop completion, the channel activates again if that channel has a minor loop channel link enabled and the link channel is itself. This effectively applies the minor loop offsets and restarts the next minor loop.
26 HALT	Halt DMA Operations

Table continues on the next page...

DMA_CR field descriptions (continued)

Field	Description
	0 Normal operation 1 Stall the start of any new channels. Executing channels are allowed to complete. Channel execution resumes when this bit is cleared.
27 HOE	Halt On Error 0 Normal operation 1 Any error causes the HALT bit to set. Subsequently, all service requests are ignored until the HALT bit is cleared.
28 ERGA	Enable Round Robin Group Arbitration 0 Fixed priority arbitration is used for selection among the groups. 1 Round robin arbitration is used for selection among the groups.
29 ERCA	Enable Round Robin Channel Arbitration 0 Fixed priority arbitration is used for channel selection within each group. 1 Round robin arbitration is used for channel selection within each group.
30 EDBG	Enable Debug 0 When in debug mode, the DMA continues to operate. 1 When in debug mode, the DMA stalls the start of a new channel. Executing channels are allowed to complete. Channel execution resumes when the system exits debug mode or the EDBG bit is cleared.
31 Reserved	This field is reserved. Reserved

16.4.6 Error Status Register (DMA_ES)

The ES provides information concerning the last recorded channel error. Channel errors can be caused by:

- A configuration error, that is:
 - An illegal setting in the transfer-control descriptor, or
 - An illegal priority register setting in fixed-arbitration
- An error termination to a bus master read or write cycle
- The UCE bit shows an uncorrectable error occurred while the device has ECC protection on the TCD SRAM
- A cancel transfer with error bit that will be set when a transfer is canceled via the corresponding cancel transfer control bit

See [Fault reporting and handling](#) for more details.

Memory map/register definition

Address: 0h base + 4h offset = 4h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	VLD	0													UCE	ECX
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	GPE	CPE	ERRCHN						SAE	SOE	DAE	DOE	NCE	SGE	SBE	DBE
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

DMA_ES field descriptions

Field	Description
0 VLD	Logical OR of all ERRH and ERRL status bits 0 No ERR bits are set. 1 At least one ERR bit is set indicating a valid error exists that has not been cleared.
1–13 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
14 UCE	Uncorrectable ECC error during channel execution. The UCE bit is set when an uncorrectable ECC error occurs on an access generated by the DMA only. If a CPU access to the TCD causes an uncorrectable ECC error, that access will receive a bus error response. NOTE: When the eDMA sees a RAM error on an IPS access (when the user is accessing a TCD), the error is reported as a bus abort. When the dma_engine (the execution engine is accessing a TCD) receives a RAM error, it is recorded in the Error Status register, DMA_ES[UCE], along with the channel number. 0 No uncorrectable ECC error 1 The last recorded error was an uncorrectable TCD RAM error
15 ECX	Transfer Canceled 0 No canceled transfers 1 The last recorded entry was a canceled transfer by the error cancel transfer input
16 GPE	Group Priority Error 0 No group priority error 1 The last recorded error was a configuration error among the group priorities. All group priorities are not unique.
17 CPE	Channel Priority Error 0 No channel priority error 1 The last recorded error was a configuration error in the channel priorities within a group. Channel priorities within a group are not unique.
18–23 ERRCHN	Error Channel Number or Canceled Channel Number The channel number of the last recorded error, excluding GPE and CPE errors, or last recorded error canceled transfer.

Table continues on the next page...

DMA_ES field descriptions (continued)

Field	Description
24 SAE	Source Address Error 0 No source address configuration error. 1 The last recorded error was a configuration error detected in the TCDn_SADDR field. TCDn_SADDR is inconsistent with TCDn_ATTR[SSIZE].
25 SOE	Source Offset Error 0 No source offset configuration error 1 The last recorded error was a configuration error detected in the TCDn_SOFF field. TCDn_SOFF is inconsistent with TCDn_ATTR[SSIZE].
26 DAE	Destination Address Error 0 No destination address configuration error 1 The last recorded error was a configuration error detected in the TCDn_DADDR field. TCDn_DADDR is inconsistent with TCDn_ATTR[DSIZE].
27 DOE	Destination Offset Error 0 No destination offset configuration error 1 The last recorded error was a configuration error detected in the TCDn_DOFF field. TCDn_DOFF is inconsistent with TCDn_ATTR[DSIZE].
28 NCE	NBYTES/CITER Configuration Error 0 No NBYTES/CITER configuration error 1 The last recorded error was a configuration error detected in the TCDn_NBYTES or TCDn_CITER fields. • TCDn_NBYTES is not a multiple of TCDn_ATTR[SSIZE] and TCDn_ATTR[DSIZE], or • TCDn_CITER[CITER] is equal to zero, or • TCDn_CITER[ELINK] is not equal to TCDn_BITER[ELINK]
29 SGE	Scatter/Gather Configuration Error 0 No scatter/gather configuration error 1 The last recorded error was a configuration error detected in the TCDn_DLASTSGA field. This field is checked at the beginning of a scatter/gather operation after major loop completion if TCDn_CSR[ESG] is enabled. TCDn_DLASTSGA is not on a 32 byte boundary.
30 SBE	Source Bus Error 0 No source bus error 1 The last recorded error was a bus error on a source read
31 DBE	Destination Bus Error 0 No destination bus error 1 The last recorded error was a bus error on a destination write

16.4.7 Enable Request Register High (DMA_ERQH)

The ERQ{H,L} registers provide a bit map for the 64 channels to enable the request signal for each channel. ERQH supports channels 63-32, while ERQL covers channels 31-00. The state of any given channel enable is directly affected by writes to this register; it is also affected by writes to the SERQ and CERQ. The {S,C}ERQ registers are provided so the request enable for a single channel can easily be modified without needing to perform a read-modify-write sequence to the ERQ{H,L}.

DMA request input signals and this enable request flag must be asserted before a channel's hardware service request is accepted. The state of the eDMA enable request flag does not affect a channel service request made explicitly through software or a linked channel request.

Address: 0h base + 8h offset = 8h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R																
W	ERQ63	ERQ62	ERQ61	ERQ60	ERQ59	ERQ58	ERQ57	ERQ56	ERQ55	ERQ54	ERQ53	ERQ52	ERQ51	ERQ50	ERQ49	ERQ48
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R																
W	ERQ47	ERQ46	ERQ45	ERQ44	ERQ43	ERQ42	ERQ41	ERQ40	ERQ39	ERQ38	ERQ37	ERQ36	ERQ35	ERQ34	ERQ33	ERQ32
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

DMA_ERQH field descriptions

Field	Description
0 ERQ63	Enable DMA Request 63 0 The DMA request signal for the corresponding channel is disabled 1 The DMA request signal for the corresponding channel is enabled
1 ERQ62	Enable DMA Request 62 0 The DMA request signal for the corresponding channel is disabled 1 The DMA request signal for the corresponding channel is enabled
2 ERQ61	Enable DMA Request 61 0 The DMA request signal for the corresponding channel is disabled 1 The DMA request signal for the corresponding channel is enabled
3 ERQ60	Enable DMA Request 60

Table continues on the next page...

DMA_ERQH field descriptions (continued)

Field	Description
	0 The DMA request signal for the corresponding channel is disabled 1 The DMA request signal for the corresponding channel is enabled
4 ERQ59	Enable DMA Request 59 0 The DMA request signal for the corresponding channel is disabled 1 The DMA request signal for the corresponding channel is enabled
5 ERQ58	Enable DMA Request 58 0 The DMA request signal for the corresponding channel is disabled 1 The DMA request signal for the corresponding channel is enabled
6 ERQ57	Enable DMA Request 57 0 The DMA request signal for the corresponding channel is disabled 1 The DMA request signal for the corresponding channel is enabled
7 ERQ56	Enable DMA Request 56 0 The DMA request signal for the corresponding channel is disabled 1 The DMA request signal for the corresponding channel is enabled
8 ERQ55	Enable DMA Request 55 0 The DMA request signal for the corresponding channel is disabled 1 The DMA request signal for the corresponding channel is enabled
9 ERQ54	Enable DMA Request 54 0 The DMA request signal for the corresponding channel is disabled 1 The DMA request signal for the corresponding channel is enabled
10 ERQ53	Enable DMA Request 53 0 The DMA request signal for the corresponding channel is disabled 1 The DMA request signal for the corresponding channel is enabled
11 ERQ52	Enable DMA Request 52 0 The DMA request signal for the corresponding channel is disabled 1 The DMA request signal for the corresponding channel is enabled
12 ERQ51	Enable DMA Request 51 0 The DMA request signal for the corresponding channel is disabled 1 The DMA request signal for the corresponding channel is enabled
13 ERQ50	Enable DMA Request 50 0 The DMA request signal for the corresponding channel is disabled 1 The DMA request signal for the corresponding channel is enabled
14 ERQ49	Enable DMA Request 49 0 The DMA request signal for the corresponding channel is disabled 1 The DMA request signal for the corresponding channel is enabled
15 ERQ48	Enable DMA Request 48

Table continues on the next page...

DMA_ERQH field descriptions (continued)

Field	Description
	0 The DMA request signal for the corresponding channel is disabled 1 The DMA request signal for the corresponding channel is enabled
16 ERQ47	Enable DMA Request 47 0 The DMA request signal for the corresponding channel is disabled 1 The DMA request signal for the corresponding channel is enabled
17 ERQ46	Enable DMA Request 46 0 The DMA request signal for the corresponding channel is disabled 1 The DMA request signal for the corresponding channel is enabled
18 ERQ45	Enable DMA Request 45 0 The DMA request signal for the corresponding channel is disabled 1 The DMA request signal for the corresponding channel is enabled
19 ERQ44	Enable DMA Request 44 0 The DMA request signal for the corresponding channel is disabled 1 The DMA request signal for the corresponding channel is enabled
20 ERQ43	Enable DMA Request 43 0 The DMA request signal for the corresponding channel is disabled 1 The DMA request signal for the corresponding channel is enabled
21 ERQ42	Enable DMA Request 42 0 The DMA request signal for the corresponding channel is disabled 1 The DMA request signal for the corresponding channel is enabled
22 ERQ41	Enable DMA Request 41 0 The DMA request signal for the corresponding channel is disabled 1 The DMA request signal for the corresponding channel is enabled
23 ERQ40	Enable DMA Request 40 0 The DMA request signal for the corresponding channel is disabled 1 The DMA request signal for the corresponding channel is enabled
24 ERQ39	Enable DMA Request 39 0 The DMA request signal for the corresponding channel is disabled 1 The DMA request signal for the corresponding channel is enabled
25 ERQ38	Enable DMA Request 38 0 The DMA request signal for the corresponding channel is disabled 1 The DMA request signal for the corresponding channel is enabled
26 ERQ37	Enable DMA Request 37 0 The DMA request signal for the corresponding channel is disabled 1 The DMA request signal for the corresponding channel is enabled
27 ERQ36	Enable DMA Request 36

Table continues on the next page...

DMA_ERQH field descriptions (continued)

Field	Description
	0 The DMA request signal for the corresponding channel is disabled 1 The DMA request signal for the corresponding channel is enabled
28 ERQ35	Enable DMA Request 35 0 The DMA request signal for the corresponding channel is disabled 1 The DMA request signal for the corresponding channel is enabled
29 ERQ34	Enable DMA Request 34 0 The DMA request signal for the corresponding channel is disabled 1 The DMA request signal for the corresponding channel is enabled
30 ERQ33	Enable DMA Request 33 0 The DMA request signal for the corresponding channel is disabled 1 The DMA request signal for the corresponding channel is enabled
31 ERQ32	Enable DMA Request 32 0 The DMA request signal for the corresponding channel is disabled 1 The DMA request signal for the corresponding channel is enabled

16.4.8 Enable Request Register Low (DMA_ERQL)

The ERQ{H,L} registers provide a bit map for the 64 channels to enable the request signal for each channel. ERQH supports channels 63-32, while ERQL covers channels 31-00. The state of any given channel enable is directly affected by writes to this register; it is also affected by writes to the SERQ and CERQ registers. These registers are provided so the request enable for a single channel can easily be modified without needing to perform a read-modify-write sequence to the ERQ{H,L}.

DMA request input signals and this enable request flag must be asserted before a channel's hardware service request is accepted. The state of the DMA enable request flag does not affect a channel service request made explicitly through software or a linked channel request.

Address: 0h base + Ch offset = Ch

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R																
W	ERQ31	ERQ30	ERQ29	ERQ28	ERQ27	ERQ26	ERQ25	ERQ24	ERQ23	ERQ22	ERQ21	ERQ20	ERQ19	ERQ18	ERQ17	ERQ16
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Memory map/register definition

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R																
W	ERQ15	ERQ14	ERQ13	ERQ12	ERQ11	ERQ10	ERQ9	ERQ8	ERQ7	ERQ6	ERQ5	ERQ4	ERQ3	ERQ2	ERQ1	ERQ0
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

DMA_ERQL field descriptions

Field	Description
0 ERQ31	Enable DMA Request 31 0 The DMA request signal for the corresponding channel is disabled 1 The DMA request signal for the corresponding channel is enabled
1 ERQ30	Enable DMA Request 30 0 The DMA request signal for the corresponding channel is disabled 1 The DMA request signal for the corresponding channel is enabled
2 ERQ29	Enable DMA Request 29 0 The DMA request signal for the corresponding channel is disabled 1 The DMA request signal for the corresponding channel is enabled
3 ERQ28	Enable DMA Request 28 0 The DMA request signal for the corresponding channel is disabled 1 The DMA request signal for the corresponding channel is enabled
4 ERQ27	Enable DMA Request 27 0 The DMA request signal for the corresponding channel is disabled 1 The DMA request signal for the corresponding channel is enabled
5 ERQ26	Enable DMA Request 26 0 The DMA request signal for the corresponding channel is disabled 1 The DMA request signal for the corresponding channel is enabled
6 ERQ25	Enable DMA Request 25 0 The DMA request signal for the corresponding channel is disabled 1 The DMA request signal for the corresponding channel is enabled
7 ERQ24	Enable DMA Request 24 0 The DMA request signal for the corresponding channel is disabled 1 The DMA request signal for the corresponding channel is enabled
8 ERQ23	Enable DMA Request 23 0 The DMA request signal for the corresponding channel is disabled 1 The DMA request signal for the corresponding channel is enabled
9 ERQ22	Enable DMA Request 22 0 The DMA request signal for the corresponding channel is disabled 1 The DMA request signal for the corresponding channel is enabled

Table continues on the next page...

DMA_ERQL field descriptions (continued)

Field	Description
10 ERQ21	Enable DMA Request 21 0 The DMA request signal for the corresponding channel is disabled 1 The DMA request signal for the corresponding channel is enabled
11 ERQ20	Enable DMA Request 20 0 The DMA request signal for the corresponding channel is disabled 1 The DMA request signal for the corresponding channel is enabled
12 ERQ19	Enable DMA Request 19 0 The DMA request signal for the corresponding channel is disabled 1 The DMA request signal for the corresponding channel is enabled
13 ERQ18	Enable DMA Request 18 0 The DMA request signal for the corresponding channel is disabled 1 The DMA request signal for the corresponding channel is enabled
14 ERQ17	Enable DMA Request 17 0 The DMA request signal for the corresponding channel is disabled 1 The DMA request signal for the corresponding channel is enabled
15 ERQ16	Enable DMA Request 16 0 The DMA request signal for the corresponding channel is disabled 1 The DMA request signal for the corresponding channel is enabled
16 ERQ15	Enable DMA Request 15 0 The DMA request signal for the corresponding channel is disabled 1 The DMA request signal for the corresponding channel is enabled
17 ERQ14	Enable DMA Request 14 0 The DMA request signal for the corresponding channel is disabled 1 The DMA request signal for the corresponding channel is enabled
18 ERQ13	Enable DMA Request 13 0 The DMA request signal for the corresponding channel is disabled 1 The DMA request signal for the corresponding channel is enabled
19 ERQ12	Enable DMA Request 12 0 The DMA request signal for the corresponding channel is disabled 1 The DMA request signal for the corresponding channel is enabled
20 ERQ11	Enable DMA Request 11 0 The DMA request signal for the corresponding channel is disabled 1 The DMA request signal for the corresponding channel is enabled
21 ERQ10	Enable DMA Request 10 0 The DMA request signal for the corresponding channel is disabled 1 The DMA request signal for the corresponding channel is enabled

Table continues on the next page...

DMA_ERQL field descriptions (continued)

Field	Description
22 ERQ9	Enable DMA Request 9 0 The DMA request signal for the corresponding channel is disabled 1 The DMA request signal for the corresponding channel is enabled
23 ERQ8	Enable DMA Request 8 0 The DMA request signal for the corresponding channel is disabled 1 The DMA request signal for the corresponding channel is enabled
24 ERQ7	Enable DMA Request 7 0 The DMA request signal for the corresponding channel is disabled 1 The DMA request signal for the corresponding channel is enabled
25 ERQ6	Enable DMA Request 6 0 The DMA request signal for the corresponding channel is disabled 1 The DMA request signal for the corresponding channel is enabled
26 ERQ5	Enable DMA Request 5 0 The DMA request signal for the corresponding channel is disabled 1 The DMA request signal for the corresponding channel is enabled
27 ERQ4	Enable DMA Request 4 0 The DMA request signal for the corresponding channel is disabled 1 The DMA request signal for the corresponding channel is enabled
28 ERQ3	Enable DMA Request 3 0 The DMA request signal for the corresponding channel is disabled 1 The DMA request signal for the corresponding channel is enabled
29 ERQ2	Enable DMA Request 2 0 The DMA request signal for the corresponding channel is disabled 1 The DMA request signal for the corresponding channel is enabled
30 ERQ1	Enable DMA Request 1 0 The DMA request signal for the corresponding channel is disabled 1 The DMA request signal for the corresponding channel is enabled
31 ERQ0	Enable DMA Request 0 0 The DMA request signal for the corresponding channel is disabled 1 The DMA request signal for the corresponding channel is enabled

16.4.9 Enable Error Interrupt Register High (DMA_EEIH)

The EEI{H,L} registers provide a bit map for the 64 channels to enable the error interrupt signal for each channel. EEIH supports channels 63-32, while EEIL covers channels 31-00. The state of any given channel's error interrupt enable is directly affected by writes to this register; it is also affected by writes to the SEEI and CEEI. The {S,C}EEI are provided so the error interrupt enable for a single channel can easily be modified without the need to perform a read-modify-write sequence to the EEI{H,L} registers.

The DMA error indicator and the error interrupt enable flag must be asserted before an error interrupt request for a given channel is asserted to the interrupt controller.

Address: 0h base + 10h offset = 10h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	EEI63	EEI62	EEI61	EEI60	EEI59	EEI58	EEI57	EEI56	EEI55	EEI54	EEI53	EEI52	EEI51	EEI50	EEI49	EEI48
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	EEI47	EEI46	EEI45	EEI44	EEI43	EEI42	EEI41	EEI40	EEI39	EEI38	EEI37	EEI36	EEI35	EEI34	EEI33	EEI32
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

DMA_EEIH field descriptions

Field	Description
0 EEI63	Enable Error Interrupt 63 0 The error signal for corresponding channel does not generate an error interrupt 1 The assertion of the error signal for corresponding channel generates an error interrupt request
1 EEI62	Enable Error Interrupt 62 0 The error signal for corresponding channel does not generate an error interrupt 1 The assertion of the error signal for corresponding channel generates an error interrupt request
2 EEI61	Enable Error Interrupt 61 0 The error signal for corresponding channel does not generate an error interrupt 1 The assertion of the error signal for corresponding channel generates an error interrupt request
3 EEI60	Enable Error Interrupt 60 0 The error signal for corresponding channel does not generate an error interrupt 1 The assertion of the error signal for corresponding channel generates an error interrupt request

Table continues on the next page...

DMA_EEIH field descriptions (continued)

Field	Description
4 EEI59	Enable Error Interrupt 59 0 The error signal for corresponding channel does not generate an error interrupt 1 The assertion of the error signal for corresponding channel generates an error interrupt request
5 EEI58	Enable Error Interrupt 58 0 The error signal for corresponding channel does not generate an error interrupt 1 The assertion of the error signal for corresponding channel generates an error interrupt request
6 EEI57	Enable Error Interrupt 57 0 The error signal for corresponding channel does not generate an error interrupt 1 The assertion of the error signal for corresponding channel generates an error interrupt request
7 EEI56	Enable Error Interrupt 56 0 The error signal for corresponding channel does not generate an error interrupt 1 The assertion of the error signal for corresponding channel generates an error interrupt request
8 EEI55	Enable Error Interrupt 55 0 The error signal for corresponding channel does not generate an error interrupt 1 The assertion of the error signal for corresponding channel generates an error interrupt request
9 EEI54	Enable Error Interrupt 54 0 The error signal for corresponding channel does not generate an error interrupt 1 The assertion of the error signal for corresponding channel generates an error interrupt request
10 EEI53	Enable Error Interrupt 53 0 The error signal for corresponding channel does not generate an error interrupt 1 The assertion of the error signal for corresponding channel generates an error interrupt request
11 EEI52	Enable Error Interrupt 52 0 The error signal for corresponding channel does not generate an error interrupt 1 The assertion of the error signal for corresponding channel generates an error interrupt request
12 EEI51	Enable Error Interrupt 51 0 The error signal for corresponding channel does not generate an error interrupt 1 The assertion of the error signal for corresponding channel generates an error interrupt request
13 EEI50	Enable Error Interrupt 50 0 The error signal for corresponding channel does not generate an error interrupt 1 The assertion of the error signal for corresponding channel generates an error interrupt request
14 EEI49	Enable Error Interrupt 49 0 The error signal for corresponding channel does not generate an error interrupt 1 The assertion of the error signal for corresponding channel generates an error interrupt request
15 EEI48	Enable Error Interrupt 48 0 The error signal for corresponding channel does not generate an error interrupt 1 The assertion of the error signal for corresponding channel generates an error interrupt request

Table continues on the next page...

DMA_EEIH field descriptions (continued)

Field	Description
16 EEI47	Enable Error Interrupt 47 0 The error signal for corresponding channel does not generate an error interrupt 1 The assertion of the error signal for corresponding channel generates an error interrupt request
17 EEI46	Enable Error Interrupt 46 0 The error signal for corresponding channel does not generate an error interrupt 1 The assertion of the error signal for corresponding channel generates an error interrupt request
18 EEI45	Enable Error Interrupt 45 0 The error signal for corresponding channel does not generate an error interrupt 1 The assertion of the error signal for corresponding channel generates an error interrupt request
19 EEI44	Enable Error Interrupt 44 0 The error signal for corresponding channel does not generate an error interrupt 1 The assertion of the error signal for corresponding channel generates an error interrupt request
20 EEI43	Enable Error Interrupt 43 0 The error signal for corresponding channel does not generate an error interrupt 1 The assertion of the error signal for corresponding channel generates an error interrupt request
21 EEI42	Enable Error Interrupt 42 0 The error signal for corresponding channel does not generate an error interrupt 1 The assertion of the error signal for corresponding channel generates an error interrupt request
22 EEI41	Enable Error Interrupt 41 0 The error signal for corresponding channel does not generate an error interrupt 1 The assertion of the error signal for corresponding channel generates an error interrupt request
23 EEI40	Enable Error Interrupt 40 0 The error signal for corresponding channel does not generate an error interrupt 1 The assertion of the error signal for corresponding channel generates an error interrupt request
24 EEI39	Enable Error Interrupt 39 0 The error signal for corresponding channel does not generate an error interrupt 1 The assertion of the error signal for corresponding channel generates an error interrupt request
25 EEI38	Enable Error Interrupt 38 0 The error signal for corresponding channel does not generate an error interrupt 1 The assertion of the error signal for corresponding channel generates an error interrupt request
26 EEI37	Enable Error Interrupt 37 0 The error signal for corresponding channel does not generate an error interrupt 1 The assertion of the error signal for corresponding channel generates an error interrupt request
27 EEI36	Enable Error Interrupt 36 0 The error signal for corresponding channel does not generate an error interrupt 1 The assertion of the error signal for corresponding channel generates an error interrupt request

Table continues on the next page...

DMA_EEIH field descriptions (continued)

Field	Description
28 EEI35	Enable Error Interrupt 35 0 The error signal for corresponding channel does not generate an error interrupt 1 The assertion of the error signal for corresponding channel generates an error interrupt request
29 EEI34	Enable Error Interrupt 34 0 The error signal for corresponding channel does not generate an error interrupt 1 The assertion of the error signal for corresponding channel generates an error interrupt request
30 EEI33	Enable Error Interrupt 33 0 The error signal for corresponding channel does not generate an error interrupt 1 The assertion of the error signal for corresponding channel generates an error interrupt request
31 EEI32	Enable Error Interrupt 32 0 The error signal for corresponding channel does not generate an error interrupt 1 The assertion of the error signal for corresponding channel generates an error interrupt request

16.4.10 Enable Error Interrupt Register Low (DMA_EEIL)

The EEIL register provides a bit map for the 64 channels to enable the error interrupt signal for each channel. EEIH supports channels 63-32, while EEIL covers channels 31-00. The state of any given channel's error interrupt enable is directly affected by writes to this register; it is also affected by writes to the SEEI and CEEI. These registers are provided so that the error interrupt enable for a single channel can easily be modified without the need to perform a read-modify-write sequence to the EEIH and EEIL registers.

The DMA error indicator and the error interrupt enable flag must be asserted before an error interrupt request for a given channel is asserted to the interrupt controller.

Address: 0h base + 14h offset = 14h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	EEI31	EEI30	EEI29	EEI28	EEI27	EEI26	EEI25	EEI24	EEI23	EEI22	EEI21	EEI20	EEI19	EEI18	EEI17	EEI16
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	EEI15	EEI14	EEI13	EEI12	EEI11	EEI10	EEI9	EEI8	EEI7	EEI6	EEI5	EEI4	EEI3	EEI2	EEI1	EEI0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

DMA_EEIL field descriptions

Field	Description
0 EEI31	Enable Error Interrupt 31 0 The error signal for corresponding channel does not generate an error interrupt 1 The assertion of the error signal for corresponding channel generates an error interrupt request
1 EEI30	Enable Error Interrupt 30 0 The error signal for corresponding channel does not generate an error interrupt 1 The assertion of the error signal for corresponding channel generates an error interrupt request
2 EEI29	Enable Error Interrupt 29 0 The error signal for corresponding channel does not generate an error interrupt 1 The assertion of the error signal for corresponding channel generates an error interrupt request
3 EEI28	Enable Error Interrupt 28 0 The error signal for corresponding channel does not generate an error interrupt 1 The assertion of the error signal for corresponding channel generates an error interrupt request
4 EEI27	Enable Error Interrupt 27 0 The error signal for corresponding channel does not generate an error interrupt 1 The assertion of the error signal for corresponding channel generates an error interrupt request
5 EEI26	Enable Error Interrupt 26 0 The error signal for corresponding channel does not generate an error interrupt 1 The assertion of the error signal for corresponding channel generates an error interrupt request
6 EEI25	Enable Error Interrupt 25 0 The error signal for corresponding channel does not generate an error interrupt 1 The assertion of the error signal for corresponding channel generates an error interrupt request
7 EEI24	Enable Error Interrupt 24 0 The error signal for corresponding channel does not generate an error interrupt 1 The assertion of the error signal for corresponding channel generates an error interrupt request
8 EEI23	Enable Error Interrupt 23 0 The error signal for corresponding channel does not generate an error interrupt 1 The assertion of the error signal for corresponding channel generates an error interrupt request
9 EEI22	Enable Error Interrupt 22 0 The error signal for corresponding channel does not generate an error interrupt 1 The assertion of the error signal for corresponding channel generates an error interrupt request
10 EEI21	Enable Error Interrupt 21 0 The error signal for corresponding channel does not generate an error interrupt 1 The assertion of the error signal for corresponding channel generates an error interrupt request
11 EEI20	Enable Error Interrupt 20 0 The error signal for corresponding channel does not generate an error interrupt 1 The assertion of the error signal for corresponding channel generates an error interrupt request

Table continues on the next page...

DMA_EEIL field descriptions (continued)

Field	Description
12 EEI19	Enable Error Interrupt 19 0 The error signal for corresponding channel does not generate an error interrupt 1 The assertion of the error signal for corresponding channel generates an error interrupt request
13 EEI18	Enable Error Interrupt 18 0 The error signal for corresponding channel does not generate an error interrupt 1 The assertion of the error signal for corresponding channel generates an error interrupt request
14 EEI17	Enable Error Interrupt 17 0 The error signal for corresponding channel does not generate an error interrupt 1 The assertion of the error signal for corresponding channel generates an error interrupt request
15 EEI16	Enable Error Interrupt 16 0 The error signal for corresponding channel does not generate an error interrupt 1 The assertion of the error signal for corresponding channel generates an error interrupt request
16 EEI15	Enable Error Interrupt 15 0 The error signal for corresponding channel does not generate an error interrupt 1 The assertion of the error signal for corresponding channel generates an error interrupt request
17 EEI14	Enable Error Interrupt 14 0 The error signal for corresponding channel does not generate an error interrupt 1 The assertion of the error signal for corresponding channel generates an error interrupt request
18 EEI13	Enable Error Interrupt 13 0 The error signal for corresponding channel does not generate an error interrupt 1 The assertion of the error signal for corresponding channel generates an error interrupt request
19 EEI12	Enable Error Interrupt 12 0 The error signal for corresponding channel does not generate an error interrupt 1 The assertion of the error signal for corresponding channel generates an error interrupt request
20 EEI11	Enable Error Interrupt 11 0 The error signal for corresponding channel does not generate an error interrupt 1 The assertion of the error signal for corresponding channel generates an error interrupt request
21 EEI10	Enable Error Interrupt 10 0 The error signal for corresponding channel does not generate an error interrupt 1 The assertion of the error signal for corresponding channel generates an error interrupt request
22 EEI9	Enable Error Interrupt 9 0 The error signal for corresponding channel does not generate an error interrupt 1 The assertion of the error signal for corresponding channel generates an error interrupt request
23 EEI8	Enable Error Interrupt 8 0 The error signal for corresponding channel does not generate an error interrupt 1 The assertion of the error signal for corresponding channel generates an error interrupt request

Table continues on the next page...

DMA_EEIL field descriptions (continued)

Field	Description
24 EEI7	Enable Error Interrupt 7 0 The error signal for corresponding channel does not generate an error interrupt 1 The assertion of the error signal for corresponding channel generates an error interrupt request
25 EEI6	Enable Error Interrupt 6 0 The error signal for corresponding channel does not generate an error interrupt 1 The assertion of the error signal for corresponding channel generates an error interrupt request
26 EEI5	Enable Error Interrupt 5 0 The error signal for corresponding channel does not generate an error interrupt 1 The assertion of the error signal for corresponding channel generates an error interrupt request
27 EEI4	Enable Error Interrupt 4 0 The error signal for corresponding channel does not generate an error interrupt 1 The assertion of the error signal for corresponding channel generates an error interrupt request
28 EEI3	Enable Error Interrupt 3 0 The error signal for corresponding channel does not generate an error interrupt 1 The assertion of the error signal for corresponding channel generates an error interrupt request
29 EEI2	Enable Error Interrupt 2 0 The error signal for corresponding channel does not generate an error interrupt 1 The assertion of the error signal for corresponding channel generates an error interrupt request
30 EEI1	Enable Error Interrupt 1 0 The error signal for corresponding channel does not generate an error interrupt 1 The assertion of the error signal for corresponding channel generates an error interrupt request
31 EEI0	Enable Error Interrupt 0 0 The error signal for corresponding channel does not generate an error interrupt 1 The assertion of the error signal for corresponding channel generates an error interrupt request

16.4.11 Set Enable Request Register (DMA_SERQ)

The SERQ provides a simple memory-mapped mechanism to set a given bit in the ERQ {H,L} to enable the DMA request for a given channel. The data value on a register write causes the corresponding bit in the ERQ {H,L} to be set. Setting the SAER bit provides a global set function, forcing the entire contents of ERQ {H,L} to be set. If the NOP bit is set, the command is ignored. This allows you to write multiple-byte registers as a 32-bit word. Reads of this register return all zeroes.

Address: 0h base + 18h offset = 18h

Bit	0	1	2	3	4	5	6	7
Read	0	0	0					
Write	NOP	SAER	SERQ					
Reset	0	0	0	0	0	0	0	0

DMA_SERQ field descriptions

Field	Description
0 NOP	No Op enable 0 Normal operation 1 No operation, ignore the other bits in this register
1 SAER	Set All Enable Requests 0 Set only the ERQ {H,L} bit specified in the SERQ field 1 Set all bits in ERQ {H,L}
2-7 SERQ	Set Enable Request Sets the corresponding bit in ERQ{H,L}.

16.4.12 Clear Enable Request Register (DMA_CERQ)

The CERQ provides a simple memory-mapped mechanism to clear a given bit in the ERQ {H,L} to disable the DMA request for a given channel. The data value on a register write causes the corresponding bit in the ERQ {H,L} to be cleared. Setting the CAER bit provides a global clear function, forcing the entire contents of the ERQ {H,L} to be cleared, disabling all DMA request inputs. If NOP is set, the command is ignored. This allows you to write multiple-byte registers as a 32-bit word. Reads of this register return all zeroes.

Address: 0h base + 19h offset = 19h

Bit	0	1	2	3	4	5	6	7
Read	0	0	0					
Write	NOP	CAER	CERQ					
Reset	0	0	0	0	0	0	0	0

DMA_CERQ field descriptions

Field	Description
0 NOP	No Op enable 0 Normal operation 1 No operation, ignore the other bits in this register
1 CAER	Clear All Enable Requests 0 Clear only the ERQ {H,L} bit specified in the CERQ field 1 Clear all bits in ERQ {H,L}
2-7 CERQ	Clear Enable Request Clears the corresponding bit in ERQ{H,L}.

16.4.13 Set Enable Error Interrupt Register (DMA_SEEI)

The SEEI provides a simple memory-mapped mechanism to set a given bit in the EEI {H,L} to enable the error interrupt for a given channel. The data value on a register write causes the corresponding bit in the EEI {H,L} to be set. Setting the SAEE bit provides a global set function, forcing the entire EEI {H,L} contents to be set. If the NOP bit is set, the command is ignored. This allows you to write multiple-byte registers as a 32-bit word. Reads of this register return all zeroes.

Address: 0h base + 1Ah offset = 1Ah

Bit	0	1	2	3	4	5	6	7
Read	0	0	0					
Write	NOP	SAEE	SEEI					
Reset	0	0	0	0	0	0	0	0

DMA_SEEI field descriptions

Field	Description
0 NOP	No Op enable 0 Normal operation 1 No operation, ignore the other bits in this register
1 SAEE	Sets All Enable Error Interrupts 0 Set only the EEI {H,L} bit specified in the SEEI field. 1 Sets all bits in EEI {H,L}
2-7 SEEI	Set Enable Error Interrupt Sets the corresponding bit in EEI {H,L}

16.4.14 Clear Enable Error Interrupt Register (DMA_CEEI)

The CEEI provides a simple memory-mapped mechanism to clear a given bit in the EEI {H,L} to disable the error interrupt for a given channel. The data value on a register write causes the corresponding bit in the EEI {H,L} to be cleared. Setting the CAEE bit provides a global clear function, forcing the EEI {H,L} contents to be cleared, disabling all DMA request inputs. If the NOP bit is set, the command is ignored. This allows you to write multiple-byte registers as a 32-bit word. Reads of this register return all zeroes.

Address: 0h base + 1Bh offset = 1Bh

Bit	0	1	2	3	4	5	6	7
Read	0	0	0					
Write	NOP	CAEE	CEEI					
Reset	0	0	0	0	0	0	0	0

DMA_CEEI field descriptions

Field	Description
0 NOP	No Op enable 0 Normal operation 1 No operation, ignore the other bits in this register
1 CAEE	Clear All Enable Error Interrupts 0 Clear only the EEI {H,L} bit specified in the CEEI field 1 Clear all bits in EEI {H,L}
2–7 CEEI	Clear Enable Error Interrupt Clears the corresponding bit in EEI{H,L}

16.4.15 Clear Interrupt Request Register (DMA_CINT)

The CINT provides a simple, memory-mapped mechanism to clear a given bit in the INT {H,L} to disable the interrupt request for a given channel. The given value on a register write causes the corresponding bit in the INT {H,L} to be cleared. Setting the CAIR bit provides a global clear function, forcing the entire contents of the INT {H,L} to be cleared, disabling all DMA interrupt requests. If the NOP bit is set, the command is ignored. This allows you to write multiple-byte registers as a 32-bit word. Reads of this register return all zeroes.

Address: 0h base + 1Ch offset = 1Ch

Bit	0	1	2	3	4	5	6	7
Read	0	0	0					
Write	NOP	CAIR	CINT					
Reset	0	0	0	0	0	0	0	0

DMA_CINT field descriptions

Field	Description
0 NOP	No Op enable 0 Normal operation 1 No operation, ignore the other bits in this register
1 CAIR	Clear All Interrupt Requests 0 Clear only the INT {H,L} bit specified in the CINT field 1 Clear all bits in INT {H,L}
2-7 CINT	Clear Interrupt Request Clears the corresponding bit in INT {H,L}

16.4.16 Clear Error Register (DMA_CERR)

The CERR provides a simple memory-mapped mechanism to clear a given bit in the ERR {H,L} to disable the error condition flag for a given channel. The given value on a register write causes the corresponding bit in the ERR {H,L} to be cleared. Setting the CAEI bit provides a global clear function, forcing the ERR {H,L} contents to be cleared, clearing all channel error indicators. If the NOP bit is set, the command is ignored. This allows you to write multiple-byte registers as a 32-bit word. Reads of this register return all zeroes.

Address: 0h base + 1Dh offset = 1Dh

Bit	0	1	2	3	4	5	6	7
Read	0	0	0					
Write	NOP	CAEI	CERR					
Reset	0	0	0	0	0	0	0	0

DMA_CERR field descriptions

Field	Description
0 NOP	No Op enable 0 Normal operation 1 No operation, ignore the other bits in this register
1 CAEI	Clear All Error Indicators 0 Clear only the ERR {H,L} bit specified in the CERR field 1 Clear all bits in ERR {H,L}
2–7 CERR	Clear Error Indicator Clears the corresponding bit in ERR {H,L}

16.4.17 Set START Bit Register (DMA_SSRT)

The SSRT provides a simple memory-mapped mechanism to set the START bit in the TCD of the given channel. The data value on a register write causes the START bit in the corresponding transfer control descriptor to be set. Setting the SAST bit provides a global set function, forcing all START bits to be set. If the NOP bit is set, the command is ignored. This allows you to write multiple-byte registers as a 32-bit word. Reads of this register return all zeroes.

Address: 0h base + 1Eh offset = 1Eh

Bit	0	1	2	3	4	5	6	7
Read	0	0	0					
Write	NOP	SAST	SSRT					
Reset	0	0	0	0	0	0	0	0

DMA_SSRT field descriptions

Field	Description
0 NOP	No Op enable 0 Normal operation 1 No operation, ignore the other bits in this register
1 SAST	Set All START Bits (activates all channels) 0 Set only the TCDn_CSR[START] bit specified in the SSRT field 1 Set all bits in TCDn_CSR[START]
2–7 SSRT	Set START Bit Sets the corresponding bit in TCDn_CSR[START]

16.4.18 Clear DONE Status Bit Register (DMA_CDNE)

The CDNE provides a simple memory-mapped mechanism to clear the DONE bit in the TCD of the given channel. The data value on a register write causes the DONE bit in the corresponding transfer control descriptor to be cleared. Setting the CADN bit provides a global clear function, forcing all DONE bits to be cleared. If the NOP bit is set, the command is ignored. This allows you to write multiple-byte registers as a 32-bit word. Reads of this register return all zeroes.

Address: 0h base + 1Fh offset = 1Fh

Bit	0	1	2	3	4	5	6	7
Read	0	0	0					
Write	NOP	CADN	CDNE					
Reset	0	0	0	0	0	0	0	0

DMA_CDNE field descriptions

Field	Description
0 NOP	No Op enable 0 Normal operation 1 No operation, ignore the other bits in this register
1 CADN	Clears All DONE Bits 0 Clears only the TCDn_CSR[DONE] bit specified in the CDNE field 1 Clears all bits in TCDn_CSR[DONE]
2–7 CDNE	Clear DONE Bit Clears the corresponding bit in TCDn_CSR[DONE]

16.4.19 Interrupt Request Register High (DMA_INTH)

The INT{H,L} provide a bit map for the 64 channels signaling the presence of an interrupt request for each channel. Depending on the appropriate bit setting in the transfer-control descriptions, the eDMA engine generates an interrupt on a data transfer completion. The outputs of this register are directly routed to the interrupt controller. During the interrupt-service routine associated with any given channel, the software must clear the appropriate bit, negating the interrupt request. Typically, a write to the CINT register in the interrupt service routine is used for this purpose.

Memory map/register definition

The state of any given channel's interrupt request is directly affected by writes to this register; it is also affected by writes to the CINT register. On writes to INT, a 1 in any bit position clears the corresponding channel's interrupt request. A zero in any bit position has no affect on the corresponding channel's current interrupt status. The CINT register is provided so the interrupt request for a single channel can easily be cleared without the need to perform a read-modify-write sequence to the INT{H,L} register.

Address: 0h base + 20h offset = 20h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	INT63	INT62	INT61	INT60	INT59	INT58	INT57	INT56	INT55	INT54	INT53	INT52	INT51	INT50	INT49	INT48
W	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	INT47	INT46	INT45	INT44	INT43	INT42	INT41	INT40	INT39	INT38	INT37	INT36	INT35	INT34	INT33	INT32
W	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

DMA_INTH field descriptions

Field	Description
0 INT63	Interrupt Request 63 0 The interrupt request for corresponding channel is cleared 1 The interrupt request for corresponding channel is active
1 INT62	Interrupt Request 62 0 The interrupt request for corresponding channel is cleared 1 The interrupt request for corresponding channel is active
2 INT61	Interrupt Request 61 0 The interrupt request for corresponding channel is cleared 1 The interrupt request for corresponding channel is active
3 INT60	Interrupt Request 60 0 The interrupt request for corresponding channel is cleared 1 The interrupt request for corresponding channel is active
4 INT59	Interrupt Request 59

Table continues on the next page...

DMA_INTH field descriptions (continued)

Field	Description
	0 The interrupt request for corresponding channel is cleared 1 The interrupt request for corresponding channel is active
5 INT58	Interrupt Request 58 0 The interrupt request for corresponding channel is cleared 1 The interrupt request for corresponding channel is active
6 INT57	Interrupt Request 57 0 The interrupt request for corresponding channel is cleared 1 The interrupt request for corresponding channel is active
7 INT56	Interrupt Request 56 0 The interrupt request for corresponding channel is cleared 1 The interrupt request for corresponding channel is active
8 INT55	Interrupt Request 55 0 The interrupt request for corresponding channel is cleared 1 The interrupt request for corresponding channel is active
9 INT54	Interrupt Request 54 0 The interrupt request for corresponding channel is cleared 1 The interrupt request for corresponding channel is active
10 INT53	Interrupt Request 53 0 The interrupt request for corresponding channel is cleared 1 The interrupt request for corresponding channel is active
11 INT52	Interrupt Request 52 0 The interrupt request for corresponding channel is cleared 1 The interrupt request for corresponding channel is active
12 INT51	Interrupt Request 51 0 The interrupt request for corresponding channel is cleared 1 The interrupt request for corresponding channel is active
13 INT50	Interrupt Request 50 0 The interrupt request for corresponding channel is cleared 1 The interrupt request for corresponding channel is active
14 INT49	Interrupt Request 49 0 The interrupt request for corresponding channel is cleared 1 The interrupt request for corresponding channel is active
15 INT48	Interrupt Request 48 0 The interrupt request for corresponding channel is cleared 1 The interrupt request for corresponding channel is active
16 INT47	Interrupt Request 47

Table continues on the next page...

DMA_INTH field descriptions (continued)

Field	Description
	0 The interrupt request for corresponding channel is cleared 1 The interrupt request for corresponding channel is active
17 INT46	Interrupt Request 46 0 The interrupt request for corresponding channel is cleared 1 The interrupt request for corresponding channel is active
18 INT45	Interrupt Request 45 0 The interrupt request for corresponding channel is cleared 1 The interrupt request for corresponding channel is active
19 INT44	Interrupt Request 44 0 The interrupt request for corresponding channel is cleared 1 The interrupt request for corresponding channel is active
20 INT43	Interrupt Request 43 0 The interrupt request for corresponding channel is cleared 1 The interrupt request for corresponding channel is active
21 INT42	Interrupt Request 42 0 The interrupt request for corresponding channel is cleared 1 The interrupt request for corresponding channel is active
22 INT41	Interrupt Request 41 0 The interrupt request for corresponding channel is cleared 1 The interrupt request for corresponding channel is active
23 INT40	Interrupt Request 40 0 The interrupt request for corresponding channel is cleared 1 The interrupt request for corresponding channel is active
24 INT39	Interrupt Request 30 0 The interrupt request for corresponding channel is cleared 1 The interrupt request for corresponding channel is active
25 INT38	Interrupt Request 38 0 The interrupt request for corresponding channel is cleared 1 The interrupt request for corresponding channel is active
26 INT37	Interrupt Request 37 0 The interrupt request for corresponding channel is cleared 1 The interrupt request for corresponding channel is active
27 INT36	Interrupt Request 36 0 The interrupt request for corresponding channel is cleared 1 The interrupt request for corresponding channel is active
28 INT35	Interrupt Request 35

Table continues on the next page...

DMA_INTH field descriptions (continued)

Field	Description
	0 The interrupt request for corresponding channel is cleared 1 The interrupt request for corresponding channel is active
29 INT34	Interrupt Request 34 0 The interrupt request for corresponding channel is cleared 1 The interrupt request for corresponding channel is active
30 INT33	Interrupt Request 33 0 The interrupt request for corresponding channel is cleared 1 The interrupt request for corresponding channel is active
31 INT32	Interrupt Request 32 0 The interrupt request for corresponding channel is cleared 1 The interrupt request for corresponding channel is active

16.4.20 Interrupt Request Register Low (DMA_INTL)

The INT {H,L} register provides a bit map for the 64 channels signaling the presence of an interrupt request for each channel. Depending on the appropriate bit setting in the transfer-control descriptors, the eDMA engine generates an interrupt on data transfer completion. The outputs of this register are directly routed to the interrupt controller. During the interrupt-service routine associated with any given channel, it is the software's responsibility to clear the appropriate bit, negating the interrupt request. Typically, a write to the CINT register in the interrupt service routine is used for this purpose.

The state of any given channel's interrupt request is directly affected by writes to this register; it is also affected by writes to the CINT register. On writes to INT, a 1 in any bit position clears the corresponding channel's interrupt request. A zero in any bit position has no affect on the corresponding channel's current interrupt status. The CINT register is provided so the interrupt request for a single channel can easily be cleared without the need to perform a read-modify-write sequence to the INT {H,L} register.

Address: 0h base + 24h offset = 24h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	INT31	INT30	INT29	INT28	INT27	INT26	INT25	INT24	INT23	INT22	INT21	INT20	INT19	INT18	INT17	INT16
W	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Memory map/register definition

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	INT15	INT14	INT13	INT12	INT11	INT10	INT9	INT8	INT7	INT6	INT5	INT4	INT3	INT2	INT1	INT0
W	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

DMA_INTL field descriptions

Field	Description
0 INT31	Interrupt Request 31 0 The interrupt request for corresponding channel is cleared 1 The interrupt request for corresponding channel is active
1 INT30	Interrupt Request 30 0 The interrupt request for corresponding channel is cleared 1 The interrupt request for corresponding channel is active
2 INT29	Interrupt Request 29 0 The interrupt request for corresponding channel is cleared 1 The interrupt request for corresponding channel is active
3 INT28	Interrupt Request 28 0 The interrupt request for corresponding channel is cleared 1 The interrupt request for corresponding channel is active
4 INT27	Interrupt Request 27 0 The interrupt request for corresponding channel is cleared 1 The interrupt request for corresponding channel is active
5 INT26	Interrupt Request 26 0 The interrupt request for corresponding channel is cleared 1 The interrupt request for corresponding channel is active
6 INT25	Interrupt Request 25 0 The interrupt request for corresponding channel is cleared 1 The interrupt request for corresponding channel is active
7 INT24	Interrupt Request 24 0 The interrupt request for corresponding channel is cleared 1 The interrupt request for corresponding channel is active
8 INT23	Interrupt Request 23 0 The interrupt request for corresponding channel is cleared 1 The interrupt request for corresponding channel is active
9 INT22	Interrupt Request 22

Table continues on the next page...

DMA_INTL field descriptions (continued)

Field	Description
	0 The interrupt request for corresponding channel is cleared 1 The interrupt request for corresponding channel is active
10 INT21	Interrupt Request 21 0 The interrupt request for corresponding channel is cleared 1 The interrupt request for corresponding channel is active
11 INT20	Interrupt Request 20 0 The interrupt request for corresponding channel is cleared 1 The interrupt request for corresponding channel is active
12 INT19	Interrupt Request 19 0 The interrupt request for corresponding channel is cleared 1 The interrupt request for corresponding channel is active
13 INT18	Interrupt Request 18 0 The interrupt request for corresponding channel is cleared 1 The interrupt request for corresponding channel is active
14 INT17	Interrupt Request 17 0 The interrupt request for corresponding channel is cleared 1 The interrupt request for corresponding channel is active
15 INT16	Interrupt Request 16 0 The interrupt request for corresponding channel is cleared 1 The interrupt request for corresponding channel is active
16 INT15	Interrupt Request 15 0 The interrupt request for corresponding channel is cleared 1 The interrupt request for corresponding channel is active
17 INT14	Interrupt Request 14 0 The interrupt request for corresponding channel is cleared 1 The interrupt request for corresponding channel is active
18 INT13	Interrupt Request 13 0 The interrupt request for corresponding channel is cleared 1 The interrupt request for corresponding channel is active
19 INT12	Interrupt Request 12 0 The interrupt request for corresponding channel is cleared 1 The interrupt request for corresponding channel is active
20 INT11	Interrupt Request 11 0 The interrupt request for corresponding channel is cleared 1 The interrupt request for corresponding channel is active
21 INT10	Interrupt Request 10

Table continues on the next page...

DMA_INTL field descriptions (continued)

Field	Description
	0 The interrupt request for corresponding channel is cleared 1 The interrupt request for corresponding channel is active
22 INT9	Interrupt Request 9 0 The interrupt request for corresponding channel is cleared 1 The interrupt request for corresponding channel is active
23 INT8	Interrupt Request 8 0 The interrupt request for corresponding channel is cleared 1 The interrupt request for corresponding channel is active
24 INT7	Interrupt Request 7 0 The interrupt request for corresponding channel is cleared 1 The interrupt request for corresponding channel is active
25 INT6	Interrupt Request 6 0 The interrupt request for corresponding channel is cleared 1 The interrupt request for corresponding channel is active
26 INT5	Interrupt Request 5 0 The interrupt request for corresponding channel is cleared 1 The interrupt request for corresponding channel is active
27 INT4	Interrupt Request 4 0 The interrupt request for corresponding channel is cleared 1 The interrupt request for corresponding channel is active
28 INT3	Interrupt Request 3 0 The interrupt request for corresponding channel is cleared 1 The interrupt request for corresponding channel is active
29 INT2	Interrupt Request 2 0 The interrupt request for corresponding channel is cleared 1 The interrupt request for corresponding channel is active
30 INT1	Interrupt Request 1 0 The interrupt request for corresponding channel is cleared 1 The interrupt request for corresponding channel is active
31 INT0	Interrupt Request 0 0 The interrupt request for corresponding channel is cleared 1 The interrupt request for corresponding channel is active

16.4.21 Error Register High (DMA_ERRH)

The ERR{H,L} provide a bit map for the 64 channels, signaling the presence of an error for each channel. ERRH supports channels 63-32, while ERRL covers channels 31-00. The eDMA engine signals the occurrence of a error condition by setting the appropriate bit in this register. The outputs of this register are enabled by the contents of EEI, then logically summed across groups of 16, 32 and 64 channels to form several group error interrupt requests. These are then routed to the interrupt controller. During the execution of the interrupt-service routine associated with any DMA errors, the software must clear the appropriate bit, negating the error-interrupt request. Typically, a write to CERR in the interrupt-service routine is used for this purpose. The normal DMA channel completion indicators, setting the transfer control descriptor DONE flag and the possible assertion of an interrupt request, are not affected when an error is detected.

The contents of this register can also be polled, because a non-zero value indicates the presence of a channel error regardless of the state of EEI. The state of any given channel's error indicators is affected by writes to this register; it is also affected by writes to CERR. On writes to ERR, a one in any bit position clears the corresponding channel's error status. A zero in any bit position has no affect on the corresponding channel's current error status. CERR is provided so the error indicator for a single channel can easily be cleared.

Address: 0h base + 28h offset = 28h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	ERR63	ERR62	ERR61	ERR60	ERR59	ERR58	ERR57	ERR56	ERR55	ERR54	ERR53	ERR52	ERR51	ERR50	ERR49	ERR48
W	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	ERR47	ERR46	ERR45	ERR44	ERR43	ERR42	ERR41	ERR40	ERR39	ERR38	ERR37	ERR36	ERR35	ERR34	ERR33	ERR32
W	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

DMA_ERRH field descriptions

Field	Description
0 ERR63	Error In Channel 63 0 An error in this channel has not occurred 1 An error in this channel has occurred
1 ERR62	Error In Channel 62 0 An error in this channel has not occurred 1 An error in this channel has occurred
2 ERR61	Error In Channel 61 0 An error in this channel has not occurred 1 An error in this channel has occurred
3 ERR60	Error In Channel 60 0 An error in this channel has not occurred 1 An error in this channel has occurred
4 ERR59	Error In Channel 59 0 An error in this channel has not occurred 1 An error in this channel has occurred
5 ERR58	Error In Channel 58 0 An error in this channel has not occurred 1 An error in this channel has occurred
6 ERR57	Error In Channel 57 0 An error in this channel has not occurred 1 An error in this channel has occurred
7 ERR56	Error In Channel 56 0 An error in this channel has not occurred 1 An error in this channel has occurred
8 ERR55	Error In Channel 55 0 An error in this channel has not occurred 1 An error in this channel has occurred
9 ERR54	Error In Channel 54 0 An error in this channel has not occurred 1 An error in this channel has occurred
10 ERR53	Error In Channel 53 0 An error in this channel has not occurred 1 An error in this channel has occurred
11 ERR52	Error In Channel 52 0 An error in this channel has not occurred 1 An error in this channel has occurred

Table continues on the next page...

DMA_ERRH field descriptions (continued)

Field	Description
12 ERR51	Error In Channel 51 0 An error in this channel has not occurred 1 An error in this channel has occurred
13 ERR50	Error In Channel 50 0 An error in this channel has not occurred 1 An error in this channel has occurred
14 ERR49	Error In Channel 49 0 An error in this channel has not occurred 1 An error in this channel has occurred
15 ERR48	Error In Channel 48 0 An error in this channel has not occurred 1 An error in this channel has occurred
16 ERR47	Error In Channel 47 0 An error in this channel has not occurred 1 An error in this channel has occurred
17 ERR46	Error In Channel 46 0 An error in this channel has not occurred 1 An error in this channel has occurred
18 ERR45	Error In Channel 45 0 An error in this channel has not occurred 1 An error in this channel has occurred
19 ERR44	Error In Channel 44 0 An error in this channel has not occurred 1 An error in this channel has occurred
20 ERR43	Error In Channel 43 0 An error in this channel has not occurred 1 An error in this channel has occurred
21 ERR42	Error In Channel 42 0 An error in this channel has not occurred 1 An error in this channel has occurred
22 ERR41	Error In Channel 41 0 An error in this channel has not occurred 1 An error in this channel has occurred
23 ERR40	Error In Channel 40 0 An error in this channel has not occurred 1 An error in this channel has occurred

Table continues on the next page...

DMA_ERRH field descriptions (continued)

Field	Description
24 ERR39	Error In Channel 39 0 An error in this channel has not occurred 1 An error in this channel has occurred
25 ERR38	Error In Channel 38 0 An error in this channel has not occurred 1 An error in this channel has occurred
26 ERR37	Error In Channel 37 0 An error in this channel has not occurred 1 An error in this channel has occurred
27 ERR36	Error In Channel 36 0 An error in this channel has not occurred 1 An error in this channel has occurred
28 ERR35	Error In Channel 35 0 An error in this channel has not occurred 1 An error in this channel has occurred
29 ERR34	Error In Channel 34 0 An error in this channel has not occurred 1 An error in this channel has occurred
30 ERR33	Error In Channel 33 0 An error in this channel has not occurred 1 An error in this channel has occurred
31 ERR32	Error In Channel 32 0 An error in this channel has not occurred 1 An error in this channel has occurred

16.4.22 Error Register Low (DMA_ERRL)

The ERR {H,L} provides a bit map for the 64 channels, signaling the presence of an error for each channel. ERRH supports channels 63-32, while ERRL covers channels 31-00. The eDMA engine signals the occurrence of an error condition by setting the appropriate bit in this register. The outputs of this register are enabled by the contents of the EEI, then logically summed across groups of 16, 32 and 64 channels to form several group error interrupt requests, which are then routed to the interrupt controller. During the execution of the interrupt-service routine associated with any DMA errors, it is software's responsibility to clear the appropriate bit, negating the error-interrupt request. Typically, a write to the CERR in the interrupt-service routine is used for this purpose. The normal DMA channel completion indicators (setting the transfer control descriptor DONE flag and the possible assertion of an interrupt request) are not affected when an error is detected.

The contents of this register can also be polled because a non-zero value indicates the presence of a channel error regardless of the state of the EEI. The state of any given channel's error indicators is affected by writes to this register; it is also affected by writes to the CERR. On writes to the ERR, a one in any bit position clears the corresponding channel's error status. A zero in any bit position has no affect on the corresponding channel's current error status. The CERR is provided so the error indicator for a single channel can easily be cleared.

Address: 0h base + 2Ch offset = 2Ch

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	ERR31	ERR30	ERR29	ERR28	ERR27	ERR26	ERR25	ERR24	ERR23	ERR22	ERR21	ERR20	ERR19	ERR18	ERR17	ERR16
W	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	ERR15	ERR14	ERR13	ERR12	ERR11	ERR10	ERR9	ERR8	ERR7	ERR6	ERR5	ERR4	ERR3	ERR2	ERR1	ERR0
W	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

DMA_ERRL field descriptions

Field	Description
0 ERR31	Error In Channel 31 0 An error in this channel has not occurred 1 An error in this channel has occurred
1 ERR30	Error In Channel 30 0 An error in this channel has not occurred 1 An error in this channel has occurred
2 ERR29	Error In Channel 29 0 An error in this channel has not occurred 1 An error in this channel has occurred
3 ERR28	Error In Channel 28 0 An error in this channel has not occurred 1 An error in this channel has occurred
4 ERR27	Error In Channel 27 0 An error in this channel has not occurred 1 An error in this channel has occurred
5 ERR26	Error In Channel 26 0 An error in this channel has not occurred 1 An error in this channel has occurred
6 ERR25	Error In Channel 25 0 An error in this channel has not occurred 1 An error in this channel has occurred
7 ERR24	Error In Channel 24 0 An error in this channel has not occurred 1 An error in this channel has occurred
8 ERR23	Error In Channel 23 0 An error in this channel has not occurred 1 An error in this channel has occurred
9 ERR22	Error In Channel 22 0 An error in this channel has not occurred 1 An error in this channel has occurred
10 ERR21	Error In Channel 21 0 An error in this channel has not occurred 1 An error in this channel has occurred
11 ERR20	Error In Channel 20 0 An error in this channel has not occurred 1 An error in this channel has occurred

Table continues on the next page...

DMA_ERRL field descriptions (continued)

Field	Description
12 ERR19	Error In Channel 19 0 An error in this channel has not occurred 1 An error in this channel has occurred
13 ERR18	Error In Channel 18 0 An error in this channel has not occurred 1 An error in this channel has occurred
14 ERR17	Error In Channel 17 0 An error in this channel has not occurred 1 An error in this channel has occurred
15 ERR16	Error In Channel 16 0 An error in this channel has not occurred 1 An error in this channel has occurred
16 ERR15	Error In Channel 15 0 An error in this channel has not occurred 1 An error in this channel has occurred
17 ERR14	Error In Channel 14 0 An error in this channel has not occurred 1 An error in this channel has occurred
18 ERR13	Error In Channel 13 0 An error in this channel has not occurred 1 An error in this channel has occurred
19 ERR12	Error In Channel 12 0 An error in this channel has not occurred 1 An error in this channel has occurred
20 ERR11	Error In Channel 11 0 An error in this channel has not occurred 1 An error in this channel has occurred
21 ERR10	Error In Channel 10 0 An error in this channel has not occurred 1 An error in this channel has occurred
22 ERR9	Error In Channel 9 0 An error in this channel has not occurred 1 An error in this channel has occurred
23 ERR8	Error In Channel 8 0 An error in this channel has not occurred 1 An error in this channel has occurred

Table continues on the next page...

DMA_ERRL field descriptions (continued)

Field	Description
24 ERR7	Error In Channel 7 0 An error in this channel has not occurred 1 An error in this channel has occurred
25 ERR6	Error In Channel 6 0 An error in this channel has not occurred 1 An error in this channel has occurred
26 ERR5	Error In Channel 5 0 An error in this channel has not occurred 1 An error in this channel has occurred
27 ERR4	Error In Channel 4 0 An error in this channel has not occurred 1 An error in this channel has occurred
28 ERR3	Error In Channel 3 0 An error in this channel has not occurred 1 An error in this channel has occurred
29 ERR2	Error In Channel 2 0 An error in this channel has not occurred 1 An error in this channel has occurred
30 ERR1	Error In Channel 1 0 An error in this channel has not occurred 1 An error in this channel has occurred
31 ERR0	Error In Channel 0 0 An error in this channel has not occurred 1 An error in this channel has occurred

16.4.23 Hardware Request Status Register High (DMA_HRSH)

The HRS{H,L} provide a bit map for the DMA channels, signaling the presence of a hardware request for each channel. HRSH supports channels 63-32, while HRSL covers channels 31-0. The hardware request status bits reflect the current state of the register and DMA request signals, qualified via the ERQ{H,L} fields, as seen by the DMA's arbitration logic. This view into the hardware request signals may be used for debug purposes.

NOTE

These bits reflect the state of the request as seen by the arbitration logic. Therefore, this status is affected by the ERQ {H,L} bits.

The HRS bit for its respective channel remains asserted for the period when a Hardware Request is Present on the Channel. After the Request is completed and Channel is free, the HRS bit is automatically cleared by hardware.

Address: 0h base + 30h offset = 30h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	HRS63	HRS62	HRS61	HRS60	HRS59	HRS58	HRS57	HRS56	HRS55	HRS54	HRS53	HRS52	HRS51	HRS50	HRS49	HRS48
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	HRS47	HRS46	HRS45	HRS44	HRS43	HRS42	HRS41	HRS40	HRS39	HRS38	HRS37	HRS36	HRS35	HRS34	HRS33	HRS32
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

DMA_HRSH field descriptions

Field	Description
0 HRS63	Hardware Request Status Channel 63 0 A hardware service request for the corresponding channel is not present 1 A hardware service request for the corresponding channel is present
1 HRS62	Hardware Request Status Channel 62 0 A hardware service request for the corresponding channel is not present 1 A hardware service request for the corresponding channel is present

Table continues on the next page...

DMA_HRSH field descriptions (continued)

Field	Description
2 HRS61	Hardware Request Status Channel 61 0 A hardware service request for the corresponding channel is not present 1 A hardware service request for the corresponding channel is present
3 HRS60	Hardware Request Status Channel 60 0 A hardware service request for the corresponding channel is not present 1 A hardware service request for the corresponding channel is present
4 HRS59	Hardware Request Status Channel 59 0 A hardware service request for the corresponding channel is not present 1 A hardware service request for the corresponding channel is present
5 HRS58	Hardware Request Status Channel 58 0 A hardware service request for the corresponding channel is not present 1 A hardware service request for the corresponding channel is present
6 HRS57	Hardware Request Status Channel 57 0 A hardware service request for the corresponding channel is not present 1 A hardware service request for the corresponding channel is present
7 HRS56	Hardware Request Status Channel 56 0 A hardware service request for the corresponding channel is not present 1 A hardware service request for the corresponding channel is present
8 HRS55	Hardware Request Status Channel 55 0 A hardware service request for the corresponding channel is not present 1 A hardware service request for the corresponding channel is present
9 HRS54	Hardware Request Status Channel 54 0 A hardware service request for the corresponding channel is not present 1 A hardware service request for the corresponding channel is present
10 HRS53	Hardware Request Status Channel 53 0 A hardware service request for the corresponding channel is not present 1 A hardware service request for the corresponding channel is present
11 HRS52	Hardware Request Status Channel 52 0 A hardware service request for the corresponding channel is not present 1 A hardware service request for the corresponding channel is present
12 HRS51	Hardware Request Status Channel 51 0 A hardware service request for the corresponding channel is not present 1 A hardware service request for the corresponding channel is present
13 HRS50	Hardware Request Status Channel 50 0 A hardware service request for the corresponding channel is not present 1 A hardware service request for the corresponding channel is present

Table continues on the next page...

DMA_HRS field descriptions (continued)

Field	Description
14 HRS49	Hardware Request Status Channel 49 0 A hardware service request for the corresponding channel is not present 1 A hardware service request for the corresponding channel is present
15 HRS48	Hardware Request Status Channel 48 0 A hardware service request for the corresponding channel is not present 1 A hardware service request for the corresponding channel is present
16 HRS47	Hardware Request Status Channel 47 0 A hardware service request for the corresponding channel is not present 1 A hardware service request for the corresponding channel is present
17 HRS46	Hardware Request Status Channel 46 0 A hardware service request for the corresponding channel is not present 1 A hardware service request for the corresponding channel is present
18 HRS45	Hardware Request Status Channel 45 0 A hardware service request for the corresponding channel is not present 1 A hardware service request for the corresponding channel is present
19 HRS44	Hardware Request Status Channel 44 0 A hardware service request for the corresponding channel is not present 1 A hardware service request for the corresponding channel is present
20 HRS43	Hardware Request Status Channel 43 0 A hardware service request for the corresponding channel is not present 1 A hardware service request for the corresponding channel is present
21 HRS42	Hardware Request Status Channel 42 0 A hardware service request for the corresponding channel is not present 1 A hardware service request for the corresponding channel is present
22 HRS41	Hardware Request Status Channel 41 0 A hardware service request for the corresponding channel is not present 1 A hardware service request for the corresponding channel is present
23 HRS40	Hardware Request Status Channel 40 0 A hardware service request for the corresponding channel is not present 1 A hardware service request for the corresponding channel is present
24 HRS39	Hardware Request Status Channel 39 0 A hardware service request for the corresponding channel is not present 1 A hardware service request for the corresponding channel is present
25 HRS38	Hardware Request Status Channel 38 0 A hardware service request for the corresponding channel is not present 1 A hardware service request for the corresponding channel is present

Table continues on the next page...

DMA_HRSH field descriptions (continued)

Field	Description
26 HRS37	Hardware Request Status Channel 37 0 A hardware service request for the corresponding channel is not present 1 A hardware service request for the corresponding channel is present
27 HRS36	Hardware Request Status Channel 36 0 A hardware service request for the corresponding channel is not present 1 A hardware service request for the corresponding channel is present
28 HRS35	Hardware Request Status Channel 35 0 A hardware service request for the corresponding channel is not present 1 A hardware service request for the corresponding channel is present
29 HRS34	Hardware Request Status Channel 34 0 A hardware service request for the corresponding channel is not present 1 A hardware service request for the corresponding channel is present
30 HRS33	Hardware Request Status Channel 33 0 A hardware service request for the corresponding channel is not present 1 A hardware service request for the corresponding channel is present
31 HRS32	Hardware Request Status Channel 32 0 A hardware service request for the corresponding channel is not present 1 A hardware service request for the corresponding channel is present

16.4.24 Hardware Request Status Register Low (DMA_HRSL)

The HRS{H,L} registers provide a bit map for the DMA channels, signaling the presence of a hardware request for each channel. HRSH supports channels 63-32, while HRSL covers channels 31-0. The hardware request status bits reflect the current state of the register and qualified (via the ERQ {H,L} fields) DMA request signals as seen by the DMA's arbitration logic. This view into the hardware request signals may be used for debug purposes.

NOTE

These bits reflect the state of the request as seen by the arbitration logic. Therefore, this status is affected by the ERQ {H,L} bits.

The HRS bit for its respective channel remains asserted for the period when a Hardware Request is Present on the Channel. After the Request is completed and Channel is free, the HRS bit is automatically cleared by hardware.

Address: 0h base + 34h offset = 34h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	HRS31	HRS30	HRS29	HRS28	HRS27	HRS26	HRS25	HRS24	HRS23	HRS22	HRS21	HRS20	HRS19	HRS18	HRS17	HRS16
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	HRS15	HRS14	HRS13	HRS12	HRS11	HRS10	HRS9	HRS8	HRS7	HRS6	HRS5	HRS4	HRS3	HRS2	HRS1	HRS0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

DMA_HRSL field descriptions

Field	Description
0 HRS31	Hardware Request Status Channel 31 0 A hardware service request for the corresponding channel is not present 1 A hardware service request for the corresponding channel is present
1 HRS30	Hardware Request Status Channel 30 0 A hardware service request for the corresponding channel is not present 1 A hardware service request for the corresponding channel is present
2 HRS29	Hardware Request Status Channel 29 0 A hardware service request for the corresponding channel is not present 1 A hardware service request for the corresponding channel is present
3 HRS28	Hardware Request Status Channel 28 0 A hardware service request for the corresponding channel is not present 1 A hardware service request for the corresponding channel is present
4 HRS27	Hardware Request Status Channel 27 0 A hardware service request for the corresponding channel is not present 1 A hardware service request for the corresponding channel is present

Table continues on the next page...

DMA_HRSL field descriptions (continued)

Field	Description
5 HRS26	Hardware Request Status Channel 26 0 A hardware service request for the corresponding channel is not present 1 A hardware service request for the corresponding channel is present
6 HRS25	Hardware Request Status Channel 25 0 A hardware service request for the corresponding channel is not present 1 A hardware service request for the corresponding channel is present
7 HRS24	Hardware Request Status Channel 24 0 A hardware service request for the corresponding channel is not present 1 A hardware service request for the corresponding channel is present
8 HRS23	Hardware Request Status Channel 23 0 A hardware service request for the corresponding channel is not present 1 A hardware service request for the corresponding channel is present
9 HRS22	Hardware Request Status Channel 22 0 A hardware service request for the corresponding channel is not present 1 A hardware service request for the corresponding channel is present
10 HRS21	Hardware Request Status Channel 21 0 A hardware service request for the corresponding channel is not present 1 A hardware service request for the corresponding channel is present
11 HRS20	Hardware Request Status Channel 20 0 A hardware service request for the corresponding channel is not present 1 A hardware service request for the corresponding channel is present
12 HRS19	Hardware Request Status Channel 19 0 A hardware service request for the corresponding channel is not present 1 A hardware service request for the corresponding channel is present
13 HRS18	Hardware Request Status Channel 18 0 A hardware service request for the corresponding channel is not present 1 A hardware service request for the corresponding channel is present
14 HRS17	Hardware Request Status Channel 17 0 A hardware service request for the corresponding channel is not present 1 A hardware service request for the corresponding channel is present
15 HRS16	Hardware Request Status Channel 16 0 A hardware service request for the corresponding channel is not present 1 A hardware service request for the corresponding channel is present
16 HRS15	Hardware Request Status Channel 15 0 A hardware service request for the corresponding channel is not present 1 A hardware service request for the corresponding channel is present

Table continues on the next page...

DMA_HRSL field descriptions (continued)

Field	Description
17 HRS14	Hardware Request Status Channel 14 0 A hardware service request for the corresponding channel is not present 1 A hardware service request for the corresponding channel is present
18 HRS13	Hardware Request Status Channel 13 0 A hardware service request for the corresponding channel is not present 1 A hardware service request for the corresponding channel is present
19 HRS12	Hardware Request Status Channel 12 0 A hardware service request for the corresponding channel is not present 1 A hardware service request for the corresponding channel is present
20 HRS11	Hardware Request Status Channel 11 0 A hardware service request for the corresponding channel is not present 1 A hardware service request for the corresponding channel is present
21 HRS10	Hardware Request Status Channel 10 0 A hardware service request for the corresponding channel is not present 1 A hardware service request for the corresponding channel is present
22 HRS9	Hardware Request Status Channel 9 0 A hardware service request for the corresponding channel is not present 1 A hardware service request for the corresponding channel is present
23 HRS8	Hardware Request Status Channel 8 0 A hardware service request for the corresponding channel is not present 1 A hardware service request for the corresponding channel is present
24 HRS7	Hardware Request Status Channel 7 0 A hardware service request for the corresponding channel is not present 1 A hardware service request for the corresponding channel is present
25 HRS6	Hardware Request Status Channel 6 0 A hardware service request for the corresponding channel is not present 1 A hardware service request for the corresponding channel is present
26 HRS5	Hardware Request Status Channel 5 0 A hardware service request for the corresponding channel is not present 1 A hardware service request for the corresponding channel is present
27 HRS4	Hardware Request Status Channel 4 0 A hardware service request for the corresponding channel is not present 1 A hardware service request for the corresponding channel is present
28 HRS3	Hardware Request Status Channel 3 0 A hardware service request for the corresponding channel is not present 1 A hardware service request for the corresponding channel is present

Table continues on the next page...

DMA_HRSL field descriptions (continued)

Field	Description
29 HRS2	Hardware Request Status Channel 2 0 A hardware service request for the corresponding channel is not present 1 A hardware service request for the corresponding channel is present
30 HRS1	Hardware Request Status Channel 1 0 A hardware service request for the corresponding channel is not present 1 A hardware service request for the corresponding channel is present
31 HRS0	Hardware Request Status Channel 0 0 A hardware service request for the corresponding channel is not present 1 A hardware service request for the corresponding channel is present

16.4.25 Global PCU Output Register (DMA_GPOR_n)

The DMAGPORH and DMAGPORL registers provides coherency information to the Peripheral Coherency Unit (PCU). The DMAGPORH and DMAGPORL registers use a bit map for the implemented channels {16,32,64} to enable a global write signal to the PCU on a per channel basis. When the bit in the corresponding channel is set, the eDMA will signal the PCU whenever that channel is performing a write. The DMAGPORH and DMAGPORL registers perform no functions within the eDMA.

NOTE

See the PCU chapter for coherency information and global write usage.

Address: 0h base + 38h offset + (4d × i), where i=0d to 1d

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R																																
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

DMA_GPOR_n field descriptions

Field	Description
0–31 GPOR	Global channel write control Setting a bit specifies the corresponding channel's writes are global.

16.4.26 Channel n Priority Register (DMA_DCHPRIn)

When fixed-priority channel arbitration is enabled ($CR[ERCA] = 0$), the contents of these registers define the unique priorities associated with each channel within a group. The channel priorities are evaluated by numeric value; for example, 0 is the lowest priority, 1 is the next higher priority, then 2, 3, etc. Software must program the channel priorities with unique values; otherwise, a configuration error is reported. The range of the priority value is limited to the values of 0 through 15. When read, the GRPPRI bits of the DCHPRIn register reflect the current priority level of the group of channels in which the corresponding channel resides. GRPPRI bits are not affected by writes to the DCHPRIn registers. The group priority is assigned in the DMA control register.

Address: $0h \text{ base} + 100h \text{ offset} + (1d \times i)$, where $i=0d$ to $63d$

Bit	0	1	2	3	4	5	6	7
Read	ECP	DPA	GRPPRI		CHPRI			
Write								
Reset	0	0	*	*	*	*	*	*

* Notes:

- CHPRI field: See bit field description.
- GRPPRI field: See bit field description.

DMA_DCHPRIn field descriptions

Field	Description
0 ECP	Enable Channel Preemption. This field resets to 0. 0 Channel n cannot be suspended by a higher priority channel's service request. 1 Channel n can be temporarily suspended by the service request of a higher priority channel.
1 DPA	Disable Preempt Ability. This field resets to 0. 0 Channel n can suspend a lower priority channel. 1 Channel n cannot suspend any channel, regardless of channel priority.
2–3 GRPPRI	Channel n Current Group Priority Group priority assigned to this channel group when fixed-priority arbitration is enabled. This field is read-only; writes are ignored. NOTE: Reset value for the group and channel priority fields, GRPPRI and CHPRI, is equal to the corresponding channel number for each priority register, that is, $DCHPRI31[GRPPRI] = 0b01$ and $DCHPRI31[CHPRI]$ equals $0b1111$.
4–7 CHPRI	Channel n Arbitration Priority Channel priority when fixed-priority arbitration is enabled

Table continues on the next page...

DMA_DCHPRIn field descriptions (continued)

Field	Description
	NOTE: Reset value for the group and channel priority fields, GRPPRI and CHPRI, is equal to the corresponding channel number for each priority register, that is, DCHPRI31[GRPPRI] = 0b01 and DCHPRI31[CHPRI] = 0b01111.

16.4.27 TCD Source Address (DMA_TCDn_SADDR)

Address: 0h base + 1000h offset + (32d × i), where i=0d to 63d

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R																																
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

DMA_TCDn_SADDR field descriptions

Field	Description
0–31 SADDR	Source Address Memory address pointing to the source data.

16.4.28 TCD Transfer Attributes (DMA_TCDn_ATTR)

Address: 0h base + 1004h offset + (32d × i), where i=0d to 63d

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Read																
Write																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

DMA_TCDn_ATTR field descriptions

Field	Description
0–4 SMOD	Source Address Modulo 0 Source address modulo feature is disabled ≠0 This value defines a specific address range specified to be the value after SADDR + SOFF calculation is performed on the original register value. Setting this field provides the ability to implement a circular data queue easily. For data queues requiring power-of-2 size bytes, the queue should start at a 0-modulo-size address and the SMOD field should be set to the appropriate value for the queue, freezing the desired number of upper address bits. The value programmed into this field specifies the number of lower address bits allowed to change. For a circular queue application, the SOFF is typically set to the transfer size to implement post-increment addressing with the SMOD function constraining the addresses to a 0-modulo-size range.
5–7 SSIZE	Source data transfer size NOTE: Using a Reserved value causes a configuration error.

Table continues on the next page...

DMA_TCDn_ATTR field descriptions (continued)

Field	Description
	000 8-bit 001 16-bit 010 32-bit 011 64-bit 100 Reserved 101 32-byte burst (4 beats of 64 bits) 110 Reserved 111 Reserved
8–12 DMOD	Destination Address Modulo See the SMOD definition
13–15 DSIZE	Destination data transfer size See the SSIZE definition

16.4.29 TCD Signed Source Address Offset (DMA_TCDn_SOFF)

Address: 0h base + 1006h offset + (32d × i), where i=0d to 63d

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Read	SOFF															
Write																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

DMA_TCDn_SOFF field descriptions

Field	Description
0–15 SOFF	Source address signed offset Sign-extended offset applied to the current source address to form the next-state value as each source read is completed.

16.4.30 TCD Minor Byte Count (Minor Loop Mapping Disabled) (DMA_TCDn_NBYTES_MLNO)

This register, or one of the next two registers (TCD_NBYTES_MLOFFNO, TCD_NBYTES_MLOFFYES), defines the number of bytes to transfer per request. Which register to use depends on whether minor loop mapping is disabled, enabled but not used for this channel, or enabled and used.

TCD word 2 is defined as follows if:

- Minor loop mapping is disabled (CR[EMLM] = 0)

If minor loop mapping is enabled, see the TCD_NBYTES_MLOFFNO and TCD_NBYTES_MLOFFYES register descriptions for the definition of TCD word 2.

Address: 0h base + 1008h offset + (32d × i), where i=0d to 63d

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	NBYTES																															
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	

DMA_TCDn_NBYTES_MLNO field descriptions

Field	Description
0–31 NBYTES	Minor Byte Transfer Count Number of bytes to be transferred in each service request of the channel. As a channel activates, the appropriate TCD contents load into the eDMA engine, and the appropriate reads and writes perform until the minor byte transfer count has transferred. This is an indivisible operation and cannot be halted. It can, however, be stalled by using the bandwidth control field, or via preemption. After the minor count is exhausted, the SADDR and DADDR values are written back into the TCD memory, the major iteration count is decremented and restored to the TCD memory. If the major iteration count is completed, additional processing is performed. NOTE: An NBYTES value of 0x0000_0000 is interpreted as a 4 GB transfer.

16.4.31 TCD Signed Minor Loop Offset (Minor Loop Mapping Enabled and Offset Disabled) (DMA_TCDn_NBYTES_MLOFFNO)

One of three registers (this register, TCD_NBYTES_MLNO, or TCD_NBYTES_MLOFFYES), defines the number of bytes to transfer per request. Which register to use depends on whether minor loop mapping is disabled, enabled but not used for this channel, or enabled and used.

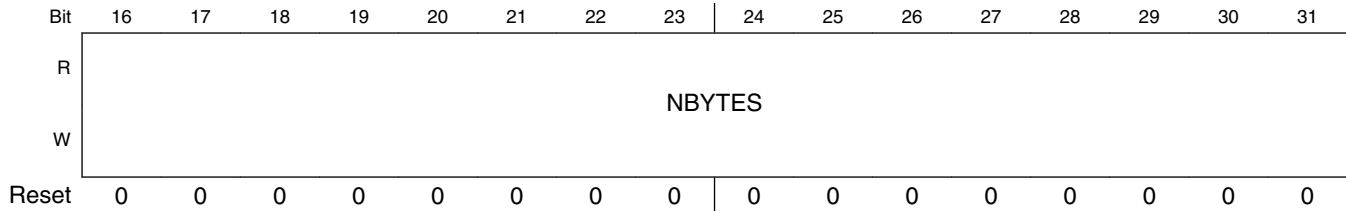
TCD word 2 is defined as follows if:

- Minor loop mapping is enabled (CR[EMLM] = 1) and
- SMLOE = 0 and DMLOE = 0

If minor loop mapping is enabled and SMLOE or DMLOE is set, then refer to the TCD_NBYTES_MLOFFYES register description. If minor loop mapping is disabled, then refer to the TCD_NBYTES_MLNO register description.

Address: 0h base + 1008h offset + (32d × i), where i=0d to 63d

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	SMLOE	DMLOE	NBYTES													
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

**DMA_TCDn_NBYTES_MLOFFNO field descriptions**

Field	Description
0 SMLOE	Source Minor Loop Offset Enable Selects whether the minor loop offset is applied to the source address upon minor loop completion. 0 The minor loop offset is not applied to the SADDR 1 The minor loop offset is applied to the SADDR
1 DMLOE	Destination Minor Loop Offset enable Selects whether the minor loop offset is applied to the destination address upon minor loop completion. 0 The minor loop offset is not applied to the DADDR 1 The minor loop offset is applied to the DADDR
2–31 NBYTES	Minor Byte Transfer Count Number of bytes to be transferred in each service request of the channel. As a channel activates, the appropriate TCD contents load into the eDMA engine, and the appropriate reads and writes perform until the minor byte transfer count has transferred. This is an indivisible operation and cannot be halted. It can, however, be stalled by using the bandwidth control field, or via preemption. After the minor count is exhausted, the SADDR and DADDR values are written back into the TCD memory, the major iteration count is decremented and restored to the TCD memory. If the major iteration count is completed, additional processing is performed.

16.4.32 TCD Signed Minor Loop Offset (Minor Loop Mapping and Offset Enabled) (DMA_TCDn_NBYTES_MLOFFYES)

One of three registers (this register, TCD_NBYTES_MLNO, or TCD_NBYTES_MLOFFNO), defines the number of bytes to transfer per request. Which register to use depends on whether minor loop mapping is disabled, enabled but not used for this channel, or enabled and used.

TCD word 2 is defined as follows if:

- Minor loop mapping is enabled (CR[EMLM] = 1) and
- Minor loop offset is enabled (SMLOE or DMLOE = 1)

If minor loop mapping is enabled and SMLOE and DMLOE are cleared, then refer to the TCD_NBYTES_MLOFFNO register description. If minor loop mapping is disabled, then refer to the TCD_NBYTES_MLNO register description.

Memory map/register definition

Address: 0h base + 1008h offset + (32d × i), where i=0d to 63d

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	SMLOE	DMLOE	MLOFF													
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	MLOFF							NBYTES								
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

DMA_TCDn_NBYTES_MLOFFYES field descriptions

Field	Description
0 SMLOE	Source Minor Loop Offset Enable Selects whether the minor loop offset is applied to the source address upon minor loop completion. 0 The minor loop offset is not applied to the SADDR 1 The minor loop offset is applied to the SADDR
1 DMLOE	Destination Minor Loop Offset enable Selects whether the minor loop offset is applied to the destination address upon minor loop completion. 0 The minor loop offset is not applied to the DADDR 1 The minor loop offset is applied to the DADDR
2–21 MLOFF	If SMLOE or DMLOE is set, this field represents a sign-extended offset applied to the source or destination address to form the next-state value after the minor loop completes.
22–31 NBYTES	Minor Byte Transfer Count Number of bytes to be transferred in each service request of the channel. As a channel activates, the appropriate TCD contents load into the eDMA engine, and the appropriate reads and writes perform until the minor byte transfer count has transferred. This is an indivisible operation and cannot be halted. It can, however, be stalled by using the bandwidth control field, or via preemption. After the minor count is exhausted, the SADDR and DADDR values are written back into the TCD memory, the major iteration count is decremented and restored to the TCD memory. If the major iteration count is completed, additional processing is performed.

16.4.33 TCD Last Source Address Adjustment (DMA_TCDn_SLAST)

Address: 0h base + 100Ch offset + (32d × i), where i=0d to 63d

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	SLAST																															
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	

DMA_TCDn_SLAST field descriptions

Field	Description
0–31 SLAST	Last Source Address Adjustment Adjustment value added to the source address at the completion of the major iteration count. This value can be applied to restore the source address to the initial value, or adjust the address to reference the next data structure. This register uses two's complement notation; the overflow bit is discarded.

16.4.34 TCD Destination Address (DMA_TCDn_DADDR)

Address: 0h base + 1010h offset + (32d × i), where i=0d to 63d

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R																																
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

DMA_TCDn_DADDR field descriptions

Field	Description
0–31 DADDR	Destination Address Memory address pointing to the destination data.

16.4.35 TCD Current Minor Loop Link, Major Loop Count (Channel Linking Enabled) (DMA_TCDn_CITER_ELINKYES)

If TCDn_CITER[ELINK] is set, the TCDn_CITER register is defined as follows.

Address: 0h base + 1014h offset + (32d × i), where i=0d to 63d

Bit	0	1	2	3	4	5	6	7
Read								
Write								
Reset	0	0	0	0	0	0	0	0

Bit	8	9	10	11	12	13	14	15
Read								
Write								
Reset	0	0	0	0	0	0	0	0

DMA_TCDn_CITER_ELINKYES field descriptions

Field	Description
0 ELINK	Enable channel-to-channel linking on minor-loop complete

Table continues on the next page...

DMA_TCDn_CITER_ELINKYES field descriptions (continued)

Field	Description
	As the channel completes the minor loop, this flag enables linking to another channel, defined by the LINKCH field. The link target channel initiates a channel service request via an internal mechanism that sets the TCDn_CSR[START] bit of the specified channel. If channel linking is disabled, the CITER value is extended to 15 bits in place of a link channel number. If the major loop is exhausted, this link mechanism is suppressed in favor of the MAJORELINK channel linking. NOTE: This bit must be equal to the BITER[ELINK] bit; otherwise, a configuration error is reported. 0 The channel-to-channel linking is disabled 1 The channel-to-channel linking is enabled
1–6 LINKCH	Minor Loop Link Channel Number If channel-to-channel linking is enabled (ELINK = 1), then after the minor loop is exhausted, the eDMA engine initiates a channel service request to the channel defined by this field by setting that channel's TCDn_CSR[START] bit.
7–15 CITER	Current Major Iteration Count This 9-bit (ELINK = 1) or 15-bit (ELINK = 0) count represents the current major loop count for the channel. It is decremented each time the minor loop is completed and updated in the transfer control descriptor memory. After the major iteration count is exhausted, the channel performs a number of operations, for example, final source and destination address calculations, optionally generating an interrupt to signal channel completion before reloading the CITER field from the Beginning Iteration Count (BITER) field. NOTE: When the CITER field is initially loaded by software, it must be set to the same value as that contained in the BITER field. NOTE: If the channel is configured to execute a single service request, the initial values of BITER and CITER should be 0x0001.

16.4.36 TCD Current Minor Loop Link, Major Loop Count (Channel Linking Disabled) (DMA_TCDn_CITER_ELINKNO)

If TCDn_CITER[ELINK] is cleared, the TCDn_CITER register is defined as follows.

Address: 0h base + 1014h offset + (32d × i), where i=0d to 63d

Bit	0	1	2	3	4	5	6	7
Read	ELINK	CITER						
Write								
Reset	0	0	0	0	0	0	0	0
Bit	8	9	10	11	12	13	14	15
Read	CITER							
Write								
Reset	0	0	0	0	0	0	0	0

DMA_TCDn_CITER_ELINKNO field descriptions

Field	Description
0 ELINK	Enable channel-to-channel linking on minor-loop complete As the channel completes the minor loop, this flag enables linking to another channel, defined by the LINKCH field. The link target channel initiates a channel service request via an internal mechanism that sets the TCDn_CSR[START] bit of the specified channel. If channel linking is disabled, the CITER value is extended to 15 bits in place of a link channel number. If the major loop is exhausted, this link mechanism is suppressed in favor of the MAJORELINK channel linking. NOTE: This bit must be equal to the BITER[ELINK] bit; otherwise, a configuration error is reported. 0 The channel-to-channel linking is disabled 1 The channel-to-channel linking is enabled
1–15 CITER	Current Major Iteration Count This 9-bit (ELINK = 1) or 15-bit (ELINK = 0) count represents the current major loop count for the channel. It is decremented each time the minor loop is completed and updated in the transfer control descriptor memory. After the major iteration count is exhausted, the channel performs a number of operations, for example, final source and destination address calculations, optionally generating an interrupt to signal channel completion before reloading the CITER field from the Beginning Iteration Count (BITER) field. NOTE: When the CITER field is initially loaded by software, it must be set to the same value as that contained in the BITER field. NOTE: If the channel is configured to execute a single service request, the initial values of BITER and CITER should be 0x0001.

16.4.37 TCD Signed Destination Address Offset (DMA_TCDn_DOFF)

Address: 0h base + 1016h offset + (32d × i), where i=0d to 63d

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Read	DOFF															
Write																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

DMA_TCDn_DOFF field descriptions

Field	Description
0–15 DOFF	Destination Address Signed Offset Sign-extended offset applied to the current destination address to form the next-state value as each destination write is completed.

16.4.38 TCD Last Destination Address Adjustment/Scatter Gather Address (DMA_TCDn_DLASTSGA)

Address: 0h base + 1018h offset + (32d × i), where i=0d to 63d

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R																																
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

DMA_TCDn_DLASTSGA field descriptions

Field	Description
0–31 DLASTSGA	Destination last address adjustment or the memory address for the next transfer control descriptor to be loaded into this channel (scatter/gather). If (TCDn_CSR[ESG] = 0) then: - Adjustment value added to the destination address at the completion of the major iteration count. This value can apply to restore the destination address to the initial value or adjust the address to reference the next data structure. - This field uses two's complement notation for the final destination address adjustment. Otherwise: - This address points to the beginning of a 0-modulo-32-byte region containing the next transfer control descriptor to be loaded into this channel. This channel reload is performed as the major iteration count completes. The scatter/gather address must be 0-modulo-32-byte, otherwise a configuration error is reported.

16.4.39 TCD Beginning Minor Loop Link, Major Loop Count (Channel Linking Enabled) (DMA_TCDn_BITER_ELINKYES)

If the TCDn_BITER[ELINK] bit is set, the TCDn_BITER register is defined as follows.

Address: 0h base + 101Ch offset + (32d × i), where i=0d to 63d

Bit	0	1	2	3	4	5	6	7
Read								
Write								
Reset	0	0	0	0	0	0	0	0

Bit	8	9	10	11	12	13	14	15
Read								
Write								
Reset	0	0	0	0	0	0	0	0

DMA_TCDn_BITER_ELINKYES field descriptions

Field	Description
0 ELINK	Enables channel-to-channel linking on minor loop complete As the channel completes the minor loop, this flag enables the linking to another channel, defined by BITER[LINKCH]. The link target channel initiates a channel service request via an internal mechanism that

Table continues on the next page...

DMA_TCDn_BITER_ELINKYES field descriptions (continued)

Field	Description
	sets the TCDn_CSR[START] bit of the specified channel. If channel linking disables, the BITER value extends to 15 bits in place of a link channel number. If the major loop is exhausted, this link mechanism is suppressed in favor of the MAJORELINK channel linking. NOTE: When the software loads the TCD, this field must be set equal to the corresponding CITER field; otherwise, a configuration error is reported. As the major iteration count is exhausted, the contents of this field are reloaded into the CITER field. 0 The channel-to-channel linking is disabled 1 The channel-to-channel linking is enabled
1–6 LINKCH	Link Channel Number If channel-to-channel linking is enabled (ELINK = 1), then after the minor loop is exhausted, the eDMA engine initiates a channel service request at the channel defined by this field by setting that channel's TCDn_CSR[START] bit. NOTE: When the software loads the TCD, this field must be set equal to the corresponding CITER field; otherwise, a configuration error is reported. As the major iteration count is exhausted, the contents of this field are reloaded into the CITER field.
7–15 BITER	Starting major iteration count As the transfer control descriptor is first loaded by software, this 9-bit (ELINK = 1) or 15-bit (ELINK = 0) field must be equal to the value in the CITER field. As the major iteration count is exhausted, the contents of this field are reloaded into the CITER field. NOTE: When the software loads the TCD, this field must be set equal to the corresponding CITER field; otherwise, a configuration error is reported. As the major iteration count is exhausted, the contents of this field are reloaded into the CITER field. If the channel is configured to execute a single service request, the initial values of BITER and CITER should be 0x0001.

16.4.40 TCD Beginning Minor Loop Link, Major Loop Count (Channel Linking Disabled) (DMA_TCDn_BITER_ELINKNO)

If the TCDn_BITER[ELINK] bit is cleared, the TCDn_BITER register is defined as follows.

Address: 0h base + 101Ch offset + (32d × i), where i=0d to 63d

Bit	0	1	2	3	4	5	6	7
Read	ELINK				BITER			
Write								
Reset	0	0	0	0	0	0	0	0
Bit	8	9	10	11	12	13	14	15
Read	BITER							
Write								
Reset	0	0	0	0	0	0	0	0

DMA_TCDn_BITER_ELINKNO field descriptions

Field	Description
0 ELINK	Enables channel-to-channel linking on minor loop complete As the channel completes the minor loop, this flag enables the linking to another channel, defined by BITER[LINKCH]. The link target channel initiates a channel service request via an internal mechanism that sets the TCDn_CSR[START] bit of the specified channel. If channel linking is disabled, the BITER value extends to 15 bits in place of a link channel number. If the major loop is exhausted, this link mechanism is suppressed in favor of the MAJORELINK channel linking. NOTE: When the software loads the TCD, this field must be set equal to the corresponding CITER field; otherwise, a configuration error is reported. As the major iteration count is exhausted, the contents of this field are reloaded into the CITER field. 0 The channel-to-channel linking is disabled 1 The channel-to-channel linking is enabled
1–15 BITER	Starting Major Iteration Count As the transfer control descriptor is first loaded by software, this 9-bit (ELINK = 1) or 15-bit (ELINK = 0) field must be equal to the value in the CITER field. As the major iteration count is exhausted, the contents of this field are reloaded into the CITER field. NOTE: When the software loads the TCD, this field must be set equal to the corresponding CITER field; otherwise, a configuration error is reported. As the major iteration count is exhausted, the contents of this field is reloaded into the CITER field. If the channel is configured to execute a single service request, the initial values of BITER and CITER should be 0x0001.

16.4.41 TCD Control and Status (DMA_TCDn_CSR)

Address: 0h base + 101Eh offset + (32d × i), where i=0d to 63d

Bit	0	1	2	3	4	5	6	7
Read	BWC			MAJORLINKCH				
Write								
Reset	0	0	0	0	0	0	0	0
Bit	8	9	10	11	12	13	14	15
Read	DONE	ACTIVE	MAJORELINK	ESG	DREQ	INTHALF	INTMAJOR	START
Write								
Reset	0	0	0	0	0	0	0	0

DMA_TCDn_CSR field descriptions

Field	Description
0–1 BWC	Bandwidth Control Throttles the amount of bus bandwidth consumed by the eDMA. Generally, as the eDMA processes the minor loop, it continuously generates read/write sequences until the minor count is exhausted. This field forces the eDMA to stall after the completion of each read/write access to control the bus request bandwidth seen by the crossbar switch.

Table continues on the next page...

DMA_TCDn_CSR field descriptions (continued)

Field	Description
	NOTE: If the source and destination sizes are equal, this field is ignored between the first and second transfers and after the last write of each minor loop. This behavior is a side effect of reducing start-up latency. 00 No eDMA engine stalls. 01 Enable eDMA master high-priority elevation (HPE) mode. No eDMA engine stalls. 10 eDMA engine stalls for 4 cycles after each R/W. 11 eDMA engine stalls for 8 cycles after each R/W.
2–7 MAJORLINKCH	Major Loop Link Channel Number If (MAJORELINK = 0) then: - No channel-to-channel linking, or chaining, is performed after the major loop counter is exhausted. Otherwise: - After the major loop counter is exhausted, the eDMA engine initiates a channel service request at the channel defined by this field by setting that channel's TCDn_CSR[START] bit.
8 DONE	Channel Done This flag indicates the eDMA has completed the major loop. The eDMA engine sets it as the CITER count reaches zero. The software clears it, or the hardware when the channel is activated. NOTE: This bit must be cleared to write the MAJORELINK or ESG bits.
9 ACTIVE	Channel Active This flag signals the channel is currently in execution. It is set when channel service begins, and is cleared by the eDMA as the minor loop completes or when any error condition is detected.
10 MAJORELINK	Enable channel-to-channel linking on major loop complete As the channel completes the major loop, this flag enables the linking to another channel, defined by MAJORLINKCH. The link target channel initiates a channel service request via an internal mechanism that sets the TCDn_CSR[START] bit of the specified channel. NOTE: To support the dynamic linking coherency model, this field is forced to zero when written to while the TCDn_CSR[DONE] bit is set. 0 The channel-to-channel linking is disabled. 1 The channel-to-channel linking is enabled.
11 ESG	Enable Scatter/Gather Processing As the channel completes the major loop, this flag enables scatter/gather processing in the current channel. If enabled, the eDMA engine uses DLASTSGA as a memory pointer to a 0-modulo-32 address containing a 32-byte data structure loaded as the transfer control descriptor into the local memory. NOTE: To support the dynamic scatter/gather coherency model, this field is forced to zero when written to while the TCDn_CSR[DONE] bit is set. 0 The current channel's TCD is normal format. 1 The current channel's TCD specifies a scatter gather format. The DLASTSGA field provides a memory pointer to the next TCD to be loaded into this channel after the major loop completes its execution.
12 DREQ	Disable Request If this flag is set, the eDMA hardware automatically clears the corresponding ERQ {H,L} bit when the current major iteration count reaches zero.

Table continues on the next page...

DMA_TCDn_CSR field descriptions (continued)

Field	Description
	0 The channel's ERQ {H,L} bit is not affected. 1 The channel's ERQ {H,L} bit is cleared when the major loop is complete.
13 INTHALF	Enable an interrupt when major counter is half complete. If this flag is set, the channel generates an interrupt request by setting the appropriate bit in the INT {H,L} register when the current major iteration count reaches the halfway point. Specifically, the comparison performed by the eDMA engine is (CITER == (BITER >> 1)). This halfway point interrupt request is provided to support double-buffered, also known as ping-pong, schemes or other types of data movement where the processor needs an early indication of the transfer's progress. NOTE: If BITER = 1, do not use INTHALF. Use INTMAJOR instead. 0 The half-point interrupt is disabled. 1 The half-point interrupt is enabled.
14 INTMAJOR	Enable an interrupt when major iteration count completes. If this flag is set, the channel generates an interrupt request by setting the appropriate bit in the INT when the current major iteration count reaches zero. 0 The end-of-major loop interrupt is disabled. 1 The end-of-major loop interrupt is enabled.
15 START	Channel Start If this flag is set, the channel is requesting service. The eDMA hardware automatically clears this flag after the channel begins execution. 0 The channel is not explicitly started. 1 The channel is explicitly started via a software initiated service request.

16.5 Functional description

The operation of the eDMA is described in the following subsections.

16.5.1 eDMA basic data flow

The basic flow of a data transfer can be partitioned into three segments.

As shown in the following diagram, the first segment involves the channel activation:

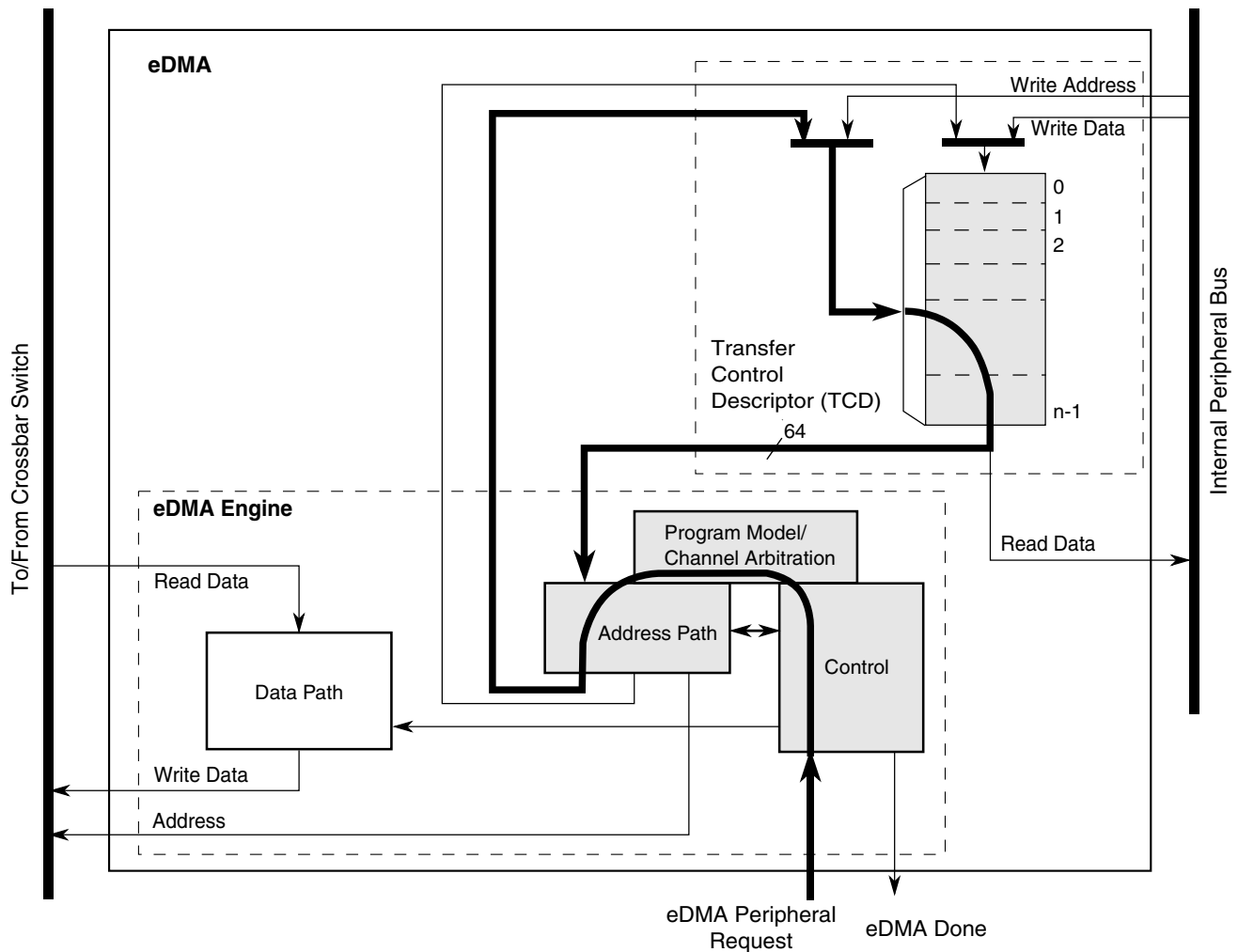


Figure 16-1065. eDMA operation, part 1

This example uses the assertion of the eDMA peripheral request signal to request service for channel n . Channel activation via software and the $\text{TCD}_n\text{-CSR}[\text{START}]$ bit follows the same basic flow as peripheral requests. The eDMA request input signal is registered internally and then routed through the eDMA engine: first through the control module, then into the program model and channel arbitration. In the next cycle, the channel arbitration performs, using the fixed-priority or round-robin algorithm. After arbitration is complete, the activated channel number is sent through the address path and converted into the required address to access the local memory for TCD_n . Next, the TCD memory is accessed and the required descriptor read from the local memory and loaded into the eDMA engine address path channel x or y registers. The TCD memory is 64 bits wide to minimize the time needed to fetch the activated channel descriptor and load it into the address path channel x or y registers.

The following diagram illustrates the second part of the basic data flow:

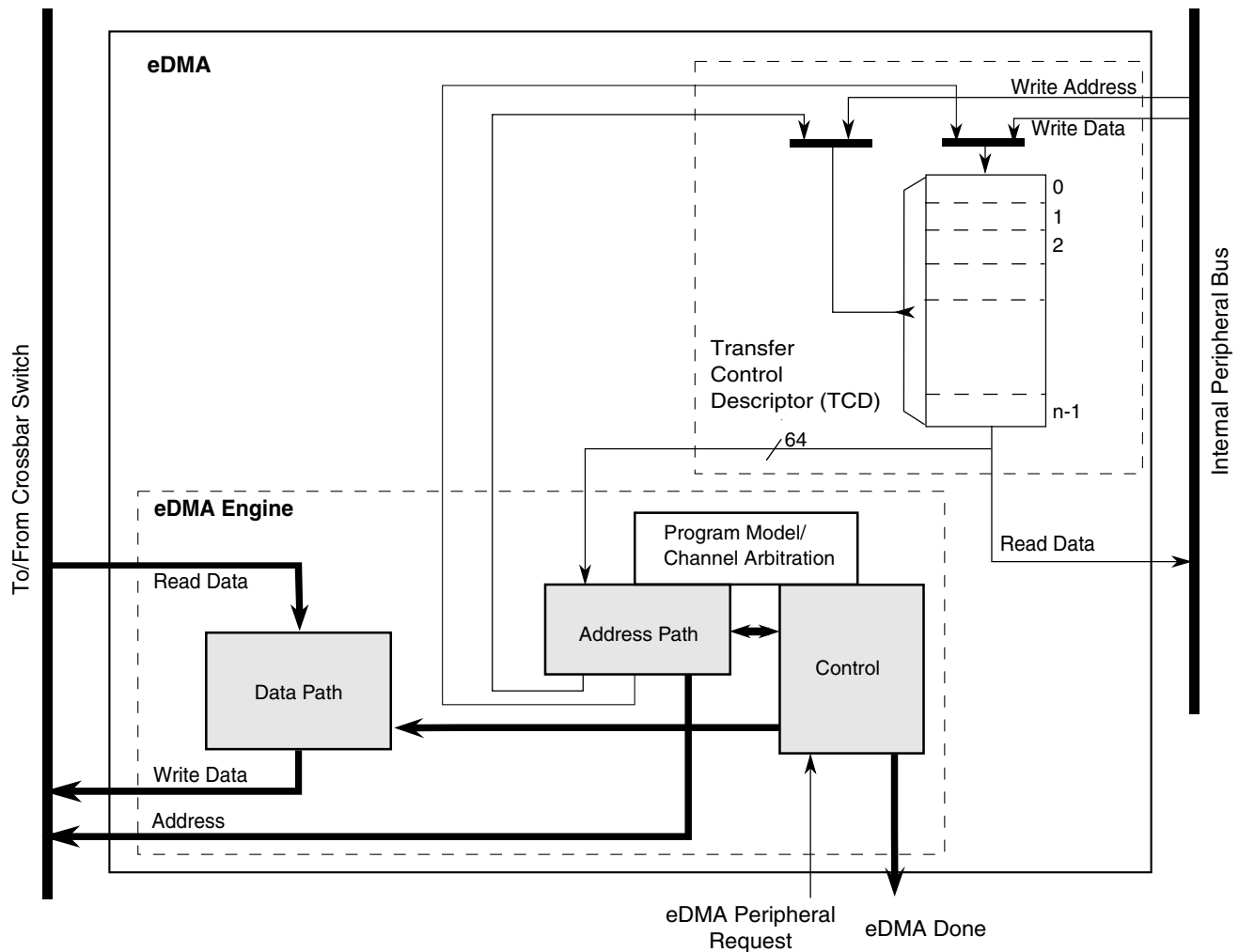


Figure 16-1066. eDMA operation, part 2

The modules associated with the data transfer (address path, data path, and control) sequence through the required source reads and destination writes to perform the actual data movement. The source reads are initiated and the fetched data is temporarily stored in the data path block until it is gated onto the internal bus during the destination write. This source read/destination write processing continues until the minor byte count has transferred.

After the minor byte count has moved, the final phase of the basic data flow is performed. In this segment, the address path logic performs the required updates to certain fields in the appropriate TCD, for example, SADDR, DADDR, CITER. If the major iteration count is exhausted, additional operations are performed. These include the final address adjustments and reloading of the BITER field into the CITER. Assertion of an optional interrupt request also occurs at this time, as does a possible fetch of a new TCD from memory using the scatter/gather address pointer included in the descriptor (if scatter/gather is enabled). The updates to the TCD memory and the assertion of an interrupt request are shown in the following diagram.

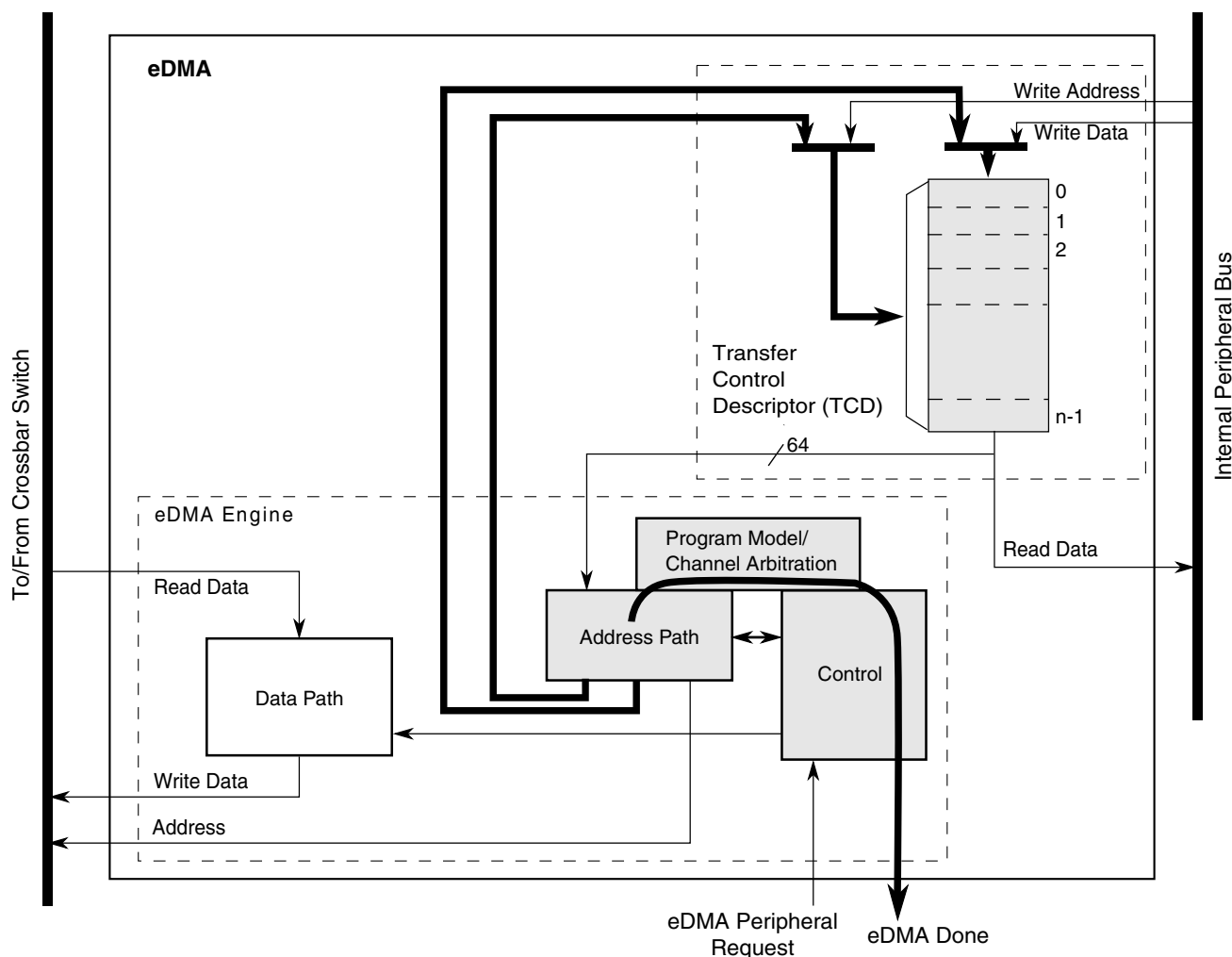


Figure 16-1067. eDMA operation, part 3

16.5.2 Fault reporting and handling

Channel errors are reported in the Error Status register (DMAx_ES) and can be caused by:

- A configuration error, which is an illegal setting in the transfer-control descriptor or an illegal priority register setting in Fixed-Arbitration mode
- An uncorrectable ECC error, or
- An error termination to a bus master read or write cycle

A configuration error is reported when the starting source or destination address, source or destination offsets, minor loop byte count, or the transfer size represent an inconsistent state. Each of these possible causes are detailed below:

- The addresses and offsets must be aligned on 0-modulo-transfer-size boundaries.

- The minor loop byte count must be a multiple of the source and destination transfer sizes.
- All source reads and destination writes must be configured to the natural boundary of the programmed transfer size respectively.
- In fixed arbitration mode, a configuration error is caused by any two channel priorities being equal within a group, or any group priority levels being equal among the groups. All channel priority levels within a group must be unique and all group priority levels among the groups must be unique when fixed arbitration mode is enabled.

NOTE

When two channels have the same priority, a channel priority error exists and will be reported in the Error Status register. However, the channel number will not be reported in the Error Status register. When all of the channel priorities within a group are not unique, the channel number selected by arbitration is undetermined.

To aid in Channel Priority Error (CPE) debug, set the Halt On Error bit in the DMA's Control Register. If all of the channel priorities within a group are not unique, the DMA will be halted after the CPE error is recorded. The DMA will remain halted and will not process any channel service requests. Once all of the channel priorities are set to unique numbers, the DMA may be enabled again by clearing the Halt bit.

- If a scatter/gather operation is enabled upon channel completion, a configuration error is reported if the scatter/gather address (DLAST_SGA) is not aligned on a 32-byte boundary.
- If minor loop channel linking is enabled upon channel completion, a configuration error is reported when the link is attempted if the TCDn_CITER[E_LINK] bit does not equal the TCDn_BITER[E_LINK] bit.

If enabled, all configuration error conditions, except the scatter/gather and minor-loop link errors, report as the channel activates and asserts an error interrupt request. A scatter/gather configuration error is reported when the scatter/gather operation begins at major loop completion when properly enabled. A minor loop channel link configuration error is reported when the link operation is serviced at minor loop completion.

If a system bus read or write is terminated with an error, the data transfer is stopped and the appropriate bus error flag set. In this case, the state of the channel's transfer control descriptor is updated by the eDMA engine with the current source address, destination address, and current iteration count at the point of the fault. When a system bus error occurs, the channel terminates after the next transfer. Due to pipeline effect, the next transfer is already in progress when the bus error is received by the eDMA. If a bus error occurs on the last read prior to beginning the write sequence, the write executes using the data captured during the bus error. If a bus error occurs on the last write prior to switching to the next read sequence, the read sequence executes before the channel terminates due to the destination bus error.

If an uncorrectable ECC error occurs during a peripheral bus access, the access is terminated with a bus error. The occurrence of an uncorrectable ECC error during an eDMA engine read causes the eDMA engine to stop the active channel immediately. The ECC error and channel number are recorded in the eDMA's Error Status register.

A transfer may be cancelled by software with the CR[CX] bit. When a cancel transfer request is recognized, the DMA engine stops processing the channel. The current read-write sequence is allowed to finish. If the cancel occurs on the last read-write sequence of a major or minor loop, the cancel request is discarded and the channel retires normally.

The error cancel transfer is the same as a cancel transfer except the Error Status register (DMAx_ES) is updated with the cancelled channel number and ECX is set. The TCD of a cancelled channel contains the source and destination addresses of the last transfer saved in the TCD. If the channel needs to be restarted, you must re-initialize the TCD because the aforementioned fields no longer represent the original parameters. When a transfer is cancelled by the error cancel transfer mechanism, the channel number is loaded into DMA_ES[ERRCHN] and ECX and VLD are set. In addition, an error interrupt may be generated if enabled.

NOTE

The cancel transfer request allows the user to stop a large data transfer in the event the full data transfer is no longer needed. The cancel transfer bit does not abort the channel. It simply stops the transferring of data and then retires the channel through its normal shutdown sequence. The application software must handle the context of the cancel. If an interrupt is desired (or not), then the interrupt should be enabled (or disabled) before the cancel request. The application software must clean up the transfer control descriptor since the full transfer did not occur.

The occurrence of any error causes the eDMA engine to stop normal processing of the active channel immediately (it goes to its error processing states and the transaction to the system bus still has pipeline effect), and the appropriate channel bit in the eDMA error register is asserted. At the same time, the details of the error condition are loaded into the Error Status register (DMAx_ES). The major loop complete indicators, setting the transfer control descriptor DONE flag and the possible assertion of an interrupt request, are not affected when an error is detected. After the error status has been updated, the eDMA engine continues operating by servicing the next appropriate channel. A channel that experiences an error condition is not automatically disabled. If a channel is terminated by an error and then issues another service request before the error is fixed, that channel executes and terminates with the same error condition.

16.5.3 Channel preemption

Channel preemption is enabled on a per-channel basis by setting the DCHPRIn[ECP] bit. Channel preemption allows the executing channel's data transfers to temporarily suspend in favor of starting a higher priority channel. After the preempting channel has completed all its minor loop data transfers, the preempted channel is restored and resumes execution. After the restored channel completes one read/write sequence, it is again eligible for preemption. If any higher priority channel is requesting service, the restored channel is suspended and the higher priority channel is serviced. Nested preemption, that is, attempting to preempt a preempting channel, is not supported. After a preempting channel begins execution, it cannot be preempted. Preemption is available only when fixed arbitration is selected for both group and channel arbitration modes.

A channel's ability to preempt another channel can be disabled by setting DCHPRIn[DPA]. When a channel's preempt ability is disabled, that channel cannot suspend a lower priority channel's data transfer, regardless of the lower priority channel's ECP setting. This allows for a pool of low priority, large data-moving channels to be defined. These low priority channels can be configured to not preempt each other, thus preventing a low priority channel from consuming the preempt slot normally available to a true, high priority channel.

16.5.4 Performance

This section addresses the performance of the eDMA module, focusing on two separate metrics:

- In the traditional data movement context, performance is best expressed as the peak data transfer rates achieved using the eDMA. In most implementations, this transfer rate is limited by the speed of the source and destination address spaces.
- In a second context where device-paced movement of single data values to/from peripherals is dominant, a measure of the requests that can be serviced in a fixed time is a more relevant metric. In this environment, the speed of the source and destination address spaces remains important. However, the microarchitecture of the eDMA also factors significantly into the resulting metric.

16.5.4.1 Peak transfer rates

The peak transfer rates for several different source and destination transfers are shown in the following tables. These tables assume:

- Internal SRAM can be accessed with zero wait-states when viewed from the system bus data phase
- All internal peripheral bus reads require two wait-states, and internal peripheral bus writes three wait-states, when viewed from the system bus data phase
- All internal peripheral bus accesses are 32-bits in size

NOTE

All architectures will not meet the assumptions listed above.
See the SRAM configuration section for more information.

This table compares peak transfer rates based on different possible system speeds. Specific chips/devices may not support all system speeds listed.

Table 16-1068. eDMA peak transfer rates (Mbytes/sec)

System Speed, Width	Internal SRAM-to-Internal SRAM	32 bit internal peripheral bus-to-Internal SRAM	Internal SRAM-to-32 bit internal peripheral bus
66.7 MHz, 64 bit	266.7	66.6	53.3
83.3 MHz, 64 bit	333.3	83.3	66.7
100.0 MHz, 64 bit	400.0	100.0	80.0
133.3 MHz, 64 bit	533.3	133.3	106.7
150.0 MHz, 64 bit	600.0	150.0	120.0

Internal-SRAM-to-internal-SRAM transfers occur at the core's datapath width. For all transfers involving the internal peripheral bus, 32-bit transfer sizes are used. In all cases, the transfer rate includes the time to read the source plus the time to write the destination.

16.5.4.2 Peak request rates

The second performance metric is a measure of the number of DMA requests that can be serviced in a given amount of time. For this metric, assume that the peripheral request causes the channel to move a single internal peripheral bus-mapped operand to/from internal SRAM. The same timing assumptions used in the previous example apply to this calculation. In particular, this metric also reflects the time required to activate the channel.

The eDMA design supports the following hardware service request sequence. Note that the exact timing from Cycle 7 is a function of the response times for the channel's read and write accesses. In the case of an internal peripheral bus read and internal SRAM write, the combined data phase time is 4 cycles. For an SRAM read and internal peripheral bus write, it is 5 cycles.

Table 16-1069. Hardware service request process

Cycle		Description
With internal peripheral bus read and internal SRAM write	With SRAM read and internal peripheral bus write	
1		eDMA peripheral request is asserted.
2		The eDMA peripheral request is registered locally in the eDMA module and qualified. TCD _n _CSR[START] bit initiated requests start at this point with the registering of the user write to TCD _n word 7.
3		Channel arbitration begins.
4		Channel arbitration completes. The transfer control descriptor local memory read is initiated.
5–6		The first two parts of the activated channel's TCD is read from the local memory. The memory width to the eDMA engine is 64 bits, so the entire descriptor can be accessed in four cycles
7		The first system bus read cycle is initiated, as the third part of the channel's TCD is read from the local memory. Depending on the state of the crossbar switch, arbitration at the system bus may insert an additional cycle of delay here.
8–11	8–12	The last part of the TCD is read in. This cycle represents the first data phase for the read, and the address phase for the destination write.
12	13	This cycle represents the data phase of the last destination write.
13	14	The eDMA engine completes the execution of the inner minor loop and prepares to write back the required TCD _n fields into the local memory. The TCD _n word 7 is read and checked for channel linking or scatter/gather requests.

Table continues on the next page...

Table 16-1069. Hardware service request process (continued)

Cycle		Description
With internal peripheral bus read and internal SRAM write	With SRAM read and internal peripheral bus write	
14	15	The appropriate fields in the first part of the TCD _n are written back into the local memory.
15	16	The fields in the second part of the TCD _n are written back into the local memory. This cycle coincides with the next channel arbitration cycle start.
16	17	The next channel to be activated performs the read of the first part of its TCD from the local memory. This is equivalent to Cycle 4 for the first channel's service request.

Assuming zero wait states on the system bus, DMA requests can be processed every 9 cycles. Assuming an average of the access times associated with internal peripheral bus-to-SRAM (4 cycles) and SRAM-to-internal peripheral bus (5 cycles), DMA requests can be processed every 11.5 cycles ($4 + (4+5)/2 + 3$). This is the time from Cycle 4 to Cycle $x + 5$. The resulting peak request rate, as a function of the system frequency, is shown in the following table.

Table 16-1070. eDMA peak request rate (MReq/sec)

System frequency (MHz)	Request rate with zero wait states	Request rate with wait states
66.6	7.4	5.8
83.3	9.2	7.2
100.0	11.1	8.7
133.3	14.8	11.6
150.0	16.6	13.0

A general formula to compute the peak request rate with overlapping requests is:

$$\text{PEAKreq} = \text{freq} / [\text{entry} + (1 + \text{read\_ws}) + (1 + \text{write\_ws}) + \text{exit}]$$

where:

Table 16-1071. Peak request formula operands

Operand	Description
PEAKreq	Peak request rate
freq	System frequency
entry	Channel startup (4 cycles)
read_ws	Wait states seen during the system bus read data phase
write_ws	Wait states seen during the system bus write data phase

Table continues on the next page...

Table 16-1071. Peak request formula operands (continued)

Operand	Description
exit	Channel shutdown (3 cycles)

16.5.4.3 eDMA performance example

Consider a system with the following characteristics:

- Internal SRAM can be accessed with one wait-state when viewed from the system bus data phase
- All internal peripheral bus reads require two wait-states, and internal peripheral bus writes three wait-states viewed from the system bus data phase
- System operates at 150 MHz

For an SRAM to internal peripheral bus transfer,

$$\text{PEAKreq} = 150 \text{ MHz} / [4 + (1 + 1) + (1 + 3) + 3] \text{ cycles} = 11.5 \text{ Mreq/sec}$$

For an internal peripheral bus to SRAM transfer,

$$\text{PEAKreq} = 150 \text{ MHz} / [4 + (1 + 2) + (1 + 1) + 3] \text{ cycles} = 12.5 \text{ Mreq/sec}$$

Assuming an even distribution of the two transfer types, the average peak request rate would be:

$$\text{PEAKreq} = (11.5 \text{ Mreq/sec} + 12.5 \text{ Mreq/sec}) / 2 = 12.0 \text{ Mreq/sec}$$

The minimum number of cycles to perform a single read/write, zero wait states on the system bus, from a cold start where no channel is executing and eDMA is idle are:

- 11 cycles for a software, that is, a `TCDn_CSR[START]` bit, request
- 12 cycles for a hardware, that is, an eDMA peripheral request signal, request

Two cycles account for the arbitration pipeline and one extra cycle on the hardware request resulting from the internal registering of the eDMA peripheral request signals. For the peak request rate calculations above, the arbitration and request registering is absorbed in or overlaps the previous executing channel.

Note

When channel linking or scatter/gather is enabled, a two cycle delay is imposed on the next channel selection and startup. This

allows the link channel or the scatter/gather channel to be eligible and considered in the arbitration pool for next channel selection.

16.6 Initialization/application information

The following sections discuss initialization of the eDMA and programming considerations.

16.6.1 eDMA initialization

To initialize the eDMA:

1. Write to the CR if a configuration other than the default is desired.
2. Write the channel priority levels to the DCHPRI n registers if a configuration other than the default is desired.
3. Enable error interrupts in the EEI registers if so desired.
4. Write the 32-byte TCD for each channel that may request service.
5. Enable any hardware service requests via the ERQH and ERQL registers.
6. Request channel service via either:
 - Software: setting the TCD n _CSR[START]
 - Hardware: slave device asserting its eDMA peripheral request signal

After any channel requests service, a channel is selected for execution based on the arbitration and priority levels written into the programmer's model. The eDMA engine reads the entire TCD, including the TCD control and status fields, as shown in the following table, for the selected channel into its internal address path module.

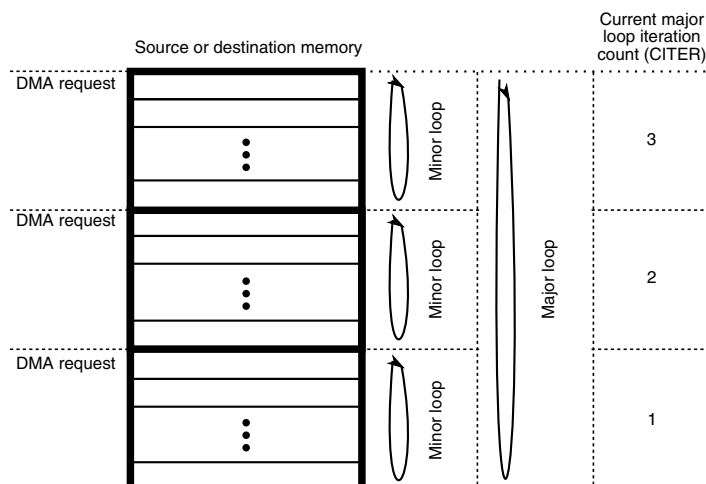
As the TCD is read, the first transfer is initiated on the internal bus, unless a configuration error is detected. Transfers from the source, as defined by TCD n _SADDR, to the destination, as defined by TCD n _DADDR, continue until the number of bytes specified by TCD n _NBYTES are transferred.

When the transfer is complete, the eDMA engine's local TCD n _SADDR, TCD n _DADDR, and TCD n _CITER are written back to the main TCD memory and any minor loop channel linking is performed, if enabled. If the major loop is exhausted, further post processing executes, such as interrupts, major loop channel linking, and scatter/gather operations, if enabled.

Table 16-1072. TCD Control and Status fields

TCDn_CSR field name	Description
START	Control bit to start channel explicitly when using a software initiated DMA service (Automatically cleared by hardware)
ACTIVE	Status bit indicating the channel is currently in execution
DONE	Status bit indicating major loop completion (cleared by software when using a software initiated DMA service)
D_REQ	Control bit to disable DMA request at end of major loop completion when using a hardware initiated DMA service
BWC	Control bits for throttling bandwidth control of a channel
E_SG	Control bit to enable scatter-gather feature
INT_HALF	Control bit to enable interrupt when major loop is half complete
INT_MAJ	Control bit to enable interrupt when major loop completes

The following figure shows how each DMA request initiates one minor-loop transfer, or iteration, without CPU intervention. DMA arbitration can occur after each minor loop, and one level of minor loop DMA preemption is allowed. The number of minor loops in a major loop is specified by the beginning iteration count (BITER).

**Figure 16-1068. Example of multiple loop iterations**

The following figure lists the memory array terms and how the TCD settings interrelate.

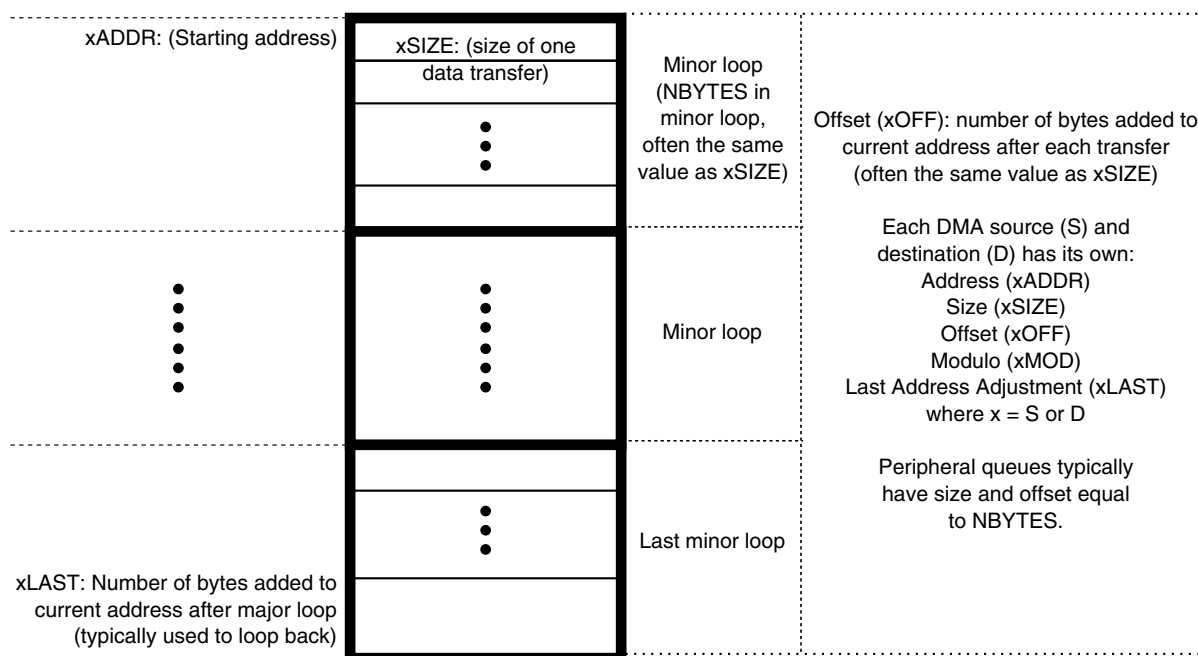


Figure 16-1069. Memory array terms

16.6.2 Programming errors

The eDMA performs various tests on the transfer control descriptor to verify consistency in the descriptor data. Most programming errors are reported on a per channel basis with the exception of two errors: Group Priority Error (ES[GPE]) and channel priority error (ES[CPE]).

For all error types other than group or channel priority errors, the channel number causing the error is recorded in the Error Status register (DMAx_ES). If the error source is not removed before the next activation of the problem channel, the error is detected and recorded again.

Channel priority errors are identified within a group once that group has been selected as the active group. For example:

1. The eDMA is configured for fixed group and fixed channel arbitration modes.
2. Group 1 is the highest priority and all channels are unique in that group.
3. Group 0 is the next highest priority and has two channels with the same priority level.
4. If Group 1 has any service requests, those requests will be executed.
5. After all of Group 1 requests have completed, Group 0 will be the next active group.

6. If Group 0 has a service request, then an undefined channel in Group 0 will be selected and a channel priority error will occur.
7. This repeats until the all of Group 0 requests have been removed or a higher priority Group 1 request comes in.

In this sequence, for item 2, the eDMA acknowledge lines will assert only if the selected channel is requesting service via the eDMA peripheral request signal. If interrupts are enabled for all channels, the user will get an error interrupt, but the channel number for the ERR register and the error interrupt request line may be wrong because they reflect the selected channel. A group priority error is global and any request in any group will cause a group priority error.

If priority levels are not unique, when any channel requests service, a channel priority error is reported. The highest channel/group priority with an active request is selected, but the lowest numbered channel/group with that priority is selected by arbitration and executed by the eDMA engine. The hardware service request handshake signals, error interrupts, and error reporting is associated with the selected channel.

16.6.3 Arbitration mode considerations

This section discusses arbitration considerations for the eDMA.

16.6.3.1 Fixed group arbitration, Fixed channel arbitration

In this mode, the channel service request from the highest priority channel in the highest priority group is selected to execute. If the eDMA is programmed so that the channels within one group use "fixed" priorities, and that group is assigned the highest "fixed" priority of all groups, that group can take all the bandwidth of the eDMA controller. That is, no other groups will be serviced if there is always at least one DMA request pending on a channel in the highest priority group when the controller arbitrates the next DMA request. The advantage of this scenario is that latency can be small for channels that need to be serviced quickly. Preemption is available in this scenario only.

16.6.3.2 Round-robin group arbitration, fixed channel arbitration

At the occurrence of one or more DMA requests from one or more groups, the channel with the highest priority from a specific group will be serviced first. Groups are serviced starting with the highest group number with an service request and rotating through to the lowest group number containing a service request.

Once the channel request is serviced, the group round robin algorithm will select the highest pending request from the next group in the round robin sequence. Servicing continues round robin, always servicing the highest priority channel in the next group in the sequence, or just skipping a group if it has no pending requests.

If a channel requests service at a rate that equals or exceeds the round robin service rate, then that channel will always be serviced before lower priority channels in the same group, and thus the lower priority channels will never be serviced. The advantage of this scenario is that no one group will consume all the eDMA bandwidth. The highest priority channel selection latency is potentially greater than fixed/fixed arbitration. Excessive request rates on high priority channels could prevent the servicing of lower priority channels in the same group.

16.6.3.3 Round-robin group arbitration, round-robin channel arbitration

Groups will be serviced as described in [Round-robin group arbitration, fixed channel arbitration](#), but this time channels will be serviced in channel number order. Only one channel is serviced from each requesting group for each round robin pass through the groups.

Within each group, channels are serviced starting with the highest channel number and rotating through to the lowest channel number without regard to channel priority levels.

Because channels are serviced in round robin manner, any channel that generates DMA requests faster than a combination of the group round robin service rate and the channel service rate for its group will not prevent the servicing of other channels in its group. Any DMA requests that are not serviced are simply lost, but at least one channel will be serviced.

This scenario ensures that all channels will be guaranteed service at some point, regardless of the request rates. However, the potential latency could be quite high. All channels are treated equally. Priority levels are not used in round robin/round robin mode.

16.6.3.4 Fixed group arbitration, Round-robin channel arbitration

The highest priority group with a request will be serviced. Lower priority groups will be serviced if no pending requests exist in the higher priority groups.

Within each group, channels are serviced starting with the highest channel number and rotating through to the lowest channel number without regard to the channel priority levels assigned within the group.

This scenario could cause the same bandwidth consumption problem as indicated in [Fixed group arbitration](#), [Fixed channel arbitration](#), but all the channels in the highest priority group will be serviced. Service latency will be short on the highest priority group, but could potentially be very much longer as the group priority decreases.

16.6.4 Performing DMA transfers

This section presents examples on how to perform DMA transfers with the eDMA.

16.6.4.1 Single request

To perform a simple transfer of n bytes of data with one activation, set the major loop to one ($\text{TCDn\_CITER} = \text{TCDn\_BITER} = 1$). The data transfer begins after the channel service request is acknowledged and the channel is selected to execute. After the transfer is complete, the $\text{TCDn\_CSR}[\text{DONE}]$ bit is set and an interrupt generates if properly enabled.

For example, the following TCD entry is configured to transfer 16 bytes of data. The eDMA is programmed for one iteration of the major loop transferring 16 bytes per iteration. The source memory has a byte wide memory port located at 0x1000. The destination memory has a 32-bit port located at 0x2000. The address offsets are programmed in increments to match the transfer size: one byte for the source and four bytes for the destination. The final source and destination addresses are adjusted to return to their beginning values.

```
TCDn_CITER = TCDn_BITER = 1
TCDn_NBYTES = 16
TCDn_SADDR = 0x1000
TCDn_SOFF = 1
TCDn_ATTR[SSIZE] = 0
TCDn_SLAST = -16
TCDn_DADDR = 0x2000
TCDn_DOFF = 4
TCDn_ATTR[DSIZE] = 2
TCDn_DLAST_SGA = -16
TCDn_CSR[INT_MAJ] = 1
TCDn_CSR[START] = 1 (Should be written last after all other fields have been initialized)
All other TCDn fields = 0
```

This generates the following event sequence:

1. User write to the $\text{TCDn\_CSR}[\text{START}]$ bit requests channel service.

2. The channel is selected by arbitration for servicing.
3. eDMA engine writes: $\text{TCDn\_CSR}[\text{DONE}] = 0$, $\text{TCDn\_CSR}[\text{START}] = 0$, $\text{TCDn\_CSR}[\text{ACTIVE}] = 1$.
4. eDMA engine reads: channel TCD data from local memory to internal register file.
5. The source-to-destination transfers are executed as follows:
 - a. Read byte from location 0x1000, read byte from location 0x1001, read byte from 0x1002, read byte from 0x1003.
 - b. Write 32-bits to location 0x2000 → first iteration of the minor loop.
 - c. Read byte from location 0x1004, read byte from location 0x1005, read byte from 0x1006, read byte from 0x1007.
 - d. Write 32-bits to location 0x2004 → second iteration of the minor loop.
 - e. Read byte from location 0x1008, read byte from location 0x1009, read byte from 0x100A, read byte from 0x100B.
 - f. Write 32-bits to location 0x2008 → third iteration of the minor loop.
 - g. Read byte from location 0x100C, read byte from location 0x100D, read byte from 0x100E, read byte from 0x100F.
 - h. Write 32-bits to location 0x200C → last iteration of the minor loop → major loop complete.
6. The eDMA engine writes: $\text{TCDn\_SADDR} = 0x1000$, $\text{TCDn\_DADDR} = 0x2000$, $\text{TCDn\_CITER} = 1$ (TCDn_BITER).
7. The eDMA engine writes: $\text{TCDn\_CSR}[\text{ACTIVE}] = 0$, $\text{TCDn\_CSR}[\text{DONE}] = 1$, $\text{INT}[n] = 1$.
8. The channel retires and the eDMA goes idle or services the next channel.

16.6.4.2 Multiple requests

The following example transfers 32 bytes via two hardware requests, but is otherwise the same as the previous example. The only fields that change are the major loop iteration count and the final address offsets. The eDMA is programmed for two iterations of the major loop transferring 16 bytes per iteration. After the channel's hardware requests are enabled in the ERQ register, the slave device initiates channel service requests.

```
TCDn_CITER = TCDn_BITER = 2  
TCDn_SLAST = -32  
TCDn_DLAST_SGA = -32
```

This would generate the following sequence of events:

1. First hardware, that is, eDMA peripheral, request for channel service.
2. The channel is selected by arbitration for servicing.
3. eDMA engine writes: $\text{TCDn\_CSR}[\text{DONE}] = 0$, $\text{TCDn\_CSR}[\text{START}] = 0$, $\text{TCDn\_CSR}[\text{ACTIVE}] = 1$.
4. eDMA engine reads: channel TCDn data from local memory to internal register file.
5. The source to destination transfers are executed as follows:
 - a. Read byte from location 0x1000, read byte from location 0x1001, read byte from 0x1002, read byte from 0x1003.
 - b. Write 32-bits to location 0x2000 → first iteration of the minor loop.
 - c. Read byte from location 0x1004, read byte from location 0x1005, read byte from 0x1006, read byte from 0x1007.
 - d. Write 32-bits to location 0x2004 → second iteration of the minor loop.
 - e. Read byte from location 0x1008, read byte from location 0x1009, read byte from 0x100A, read byte from 0x100B.
 - f. Write 32-bits to location 0x2008 → third iteration of the minor loop.
 - g. Read byte from location 0x100C, read byte from location 0x100D, read byte from 0x100E, read byte from 0x100F.
 - h. Write 32-bits to location 0x200C → last iteration of the minor loop.
6. eDMA engine writes: $\text{TCDn\_SADDR} = 0x1010$, $\text{TCDn\_DADDR} = 0x2010$, $\text{TCDn\_CITER} = 1$.
7. eDMA engine writes: $\text{TCDn\_CSR}[\text{ACTIVE}] = 0$.
8. The channel retires → one iteration of the major loop. The eDMA goes idle or services the next channel.
9. Second hardware, that is, eDMA peripheral, requests channel service.
10. The channel is selected by arbitration for servicing.
11. eDMA engine writes: $\text{TCDn\_CSR}[\text{DONE}] = 0$, $\text{TCDn\_CSR}[\text{START}] = 0$, $\text{TCDn\_CSR}[\text{ACTIVE}] = 1$.

12. eDMA engine reads: channel TCD data from local memory to internal register file.
13. The source to destination transfers are executed as follows:
 - a. Read byte from location 0x1010, read byte from location 0x1011, read byte from 0x1012, read byte from 0x1013.
 - b. Write 32-bits to location 0x2010 → first iteration of the minor loop.
 - c. Read byte from location 0x1014, read byte from location 0x1015, read byte from 0x1016, read byte from 0x1017.
 - d. Write 32-bits to location 0x2014 → second iteration of the minor loop.
 - e. Read byte from location 0x1018, read byte from location 0x1019, read byte from 0x101A, read byte from 0x101B.
 - f. Write 32-bits to location 0x2018 → third iteration of the minor loop.
 - g. Read byte from location 0x101C, read byte from location 0x101D, read byte from 0x101E, read byte from 0x101F.
 - h. Write 32-bits to location 0x201C → last iteration of the minor loop → major loop complete.
14. eDMA engine writes: $\text{TCDn\_SADDR} = 0x1000$, $\text{TCDn\_DADDR} = 0x2000$, $\text{TCDn\_CITER} = 2$ (TCDn_BITER).
15. eDMA engine writes: $\text{TCDn\_CSR}[\text{ACTIVE}] = 0$, $\text{TCDn\_CSR}[\text{DONE}] = 1$, $\text{INT}[n] = 1$.
16. The channel retires → major loop complete. The eDMA goes idle or services the next channel.

16.6.4.3 Using the modulo feature

The modulo feature of the eDMA provides the ability to implement a circular data queue in which the size of the queue is a power of 2. MOD is a 5-bit field for the source and destination in the TCD, and it specifies which lower address bits increment from their original value after the address+offset calculation. All upper address bits remain the same as in the original value. A setting of 0 for this field disables the modulo feature.

The following table shows how the transfer addresses are specified based on the setting of the MOD field. Here a circular buffer is created where the address wraps to the original value while the 28 upper address bits (0x1234567x) retain their original value. In this example the source address is set to 0x12345670, the offset is set to 4 bytes and the MOD field is set to 4, allowing for a 2⁴ byte (16-byte) size queue.

Table 16-1073. Modulo example

Transfer Number	Address
1	0x12345670
2	0x12345674
3	0x12345678
4	0x1234567C
5	0x12345670
6	0x12345674

16.6.5 Monitoring transfer descriptor status

This section discusses how to monitor eDMA status.

16.6.5.1 Testing for minor loop completion

There are two methods to test for minor loop completion when using software initiated service requests. The first is to read the TCD_n_CITER field and test for a change. Another method may be extracted from the sequence shown below. The second method is to test the TCD_n_CSR[START] bit and the TCD_n_CSR[ACTIVE] bit. The minor-loop-complete condition is indicated by both bits reading zero after the TCD_n_CSR[START] was set. Polling the TCD_n_CSR[ACTIVE] bit may be inconclusive, because the active status may be missed if the channel execution is short in duration.

The TCD status bits execute the following sequence for a software activated channel:

Stage	TCD _n _CSR bits			State
	START	ACTIVE	DONE	
1	1	0	0	Channel service request via software
2	0	1	0	Channel is executing
3a	0	0	0	Channel has completed the minor loop and is idle
3b	0	0	1	Channel has completed the major loop and is idle

The best method to test for minor-loop completion when using hardware, that is, peripheral, initiated service requests is to read the `TCDn_CITER` field and test for a change. The hardware request and acknowledge handshake signals are not visible in the programmer's model.

The TCD status bits execute the following sequence for a hardware-activated channel:

Stage	TCDn_CSR bits			State
	START	ACTIVE	DONE	
1	0	0	0	Channel service request via hardware (peripheral request asserted)
2	0	1	0	Channel is executing
3a	0	0	0	Channel has completed the minor loop and is idle
3b	0	0	1	Channel has completed the major loop and is idle

For both activation types, the major-loop-complete status is explicitly indicated via the `TCDn_CSR[DONE]` bit.

The `TCDn_CSR[START]` bit is cleared automatically when the channel begins execution regardless of how the channel activates.

16.6.5.2 Reading the transfer descriptors of active channels

The eDMA reads back the true `TCDn_SADDR`, `TCDn_DADDR`, and `TCDn_NBYTES` values if read while a channel executes. The true values of the `SADDR`, `DADDR`, and `NBYTES` are the values the eDMA engine currently uses in its internal register file and not the values in the TCD local memory for that channel. The addresses, `SADDR` and `DADDR`, and `NBYTES`, which decrement to zero as the transfer progresses, can give an indication of the progress of the transfer. All other values are read back from the TCD local memory.

16.6.5.3 Checking channel preemption status

Preemption is available only when fixed arbitration is selected for both group and channel arbitration modes. A preemptive situation is one in which a preempt-enabled channel runs and a higher priority request becomes active. When the eDMA engine is not operating in fixed group, fixed channel arbitration mode, the determination of the actively running relative priority outstanding requests become undefined. Channel and/or group priorities are treated as equal, that is, constantly rotating, when Round-Robin Arbitration mode is selected.

The TCD n _CSR[ACTIVE] bit for the preempted channel remains asserted throughout the preemption. The preempted channel is temporarily suspended while the preempting channel executes one major loop iteration. If two TCD n _CSR[ACTIVE] bits are set simultaneously in the global TCD map, a higher priority channel is actively preempting a lower priority channel.

16.6.6 Channel Linking

Channel linking (or chaining) is a mechanism where one channel sets the TCD n _CSR[START] bit of another channel (or itself), therefore initiating a service request for that channel. When properly enabled, the EDMA engine automatically performs this operation at the major or minor loop completion.

The minor loop channel linking occurs at the completion of the minor loop (or one iteration of the major loop). The TCD n _CITER[E_LINK] field determines whether a minor loop link is requested. When enabled, the channel link is made after each iteration of the major loop except for the last. When the major loop is exhausted, only the major loop channel link fields are used to determine if a channel link should be made. For example, the initial fields of:

```
TCDn_CITER[E_LINK] = 1
TCDn_CITER[LINKCH] = 0xC
TCDn_CITER[CITER] value = 0x4
TCDn_CSR[MAJOR_E_LINK] = 1
TCDn_CSR[MAJOR_LINKCH] = 0x7
```

executes as:

1. Minor loop done → set TCD12_CSR[START] bit
2. Minor loop done → set TCD12_CSR[START] bit
3. Minor loop done → set TCD12_CSR[START] bit
4. Minor loop done, major loop done → set TCD7_CSR[START] bit

When minor loop linking is enabled (TCD n _CITER[E_LINK] = 1), the TCD n _CITER[CITER] field uses a nine bit vector to form the current iteration count. When minor loop linking is disabled (TCD n _CITER[E_LINK] = 0), the TCD n _CITER[CITER] field uses a 15-bit vector to form the current iteration count. The bits associated with the TCD n _CITER[LINKCH] field are concatenated onto the CITER value to increase the range of the CITER.

Note

The TCD_n_CITER[E_LINK] bit and the TCD_n_BITER[E_LINK] bit must equal or a configuration error is reported. The CITER and BITER vector widths must be equal to calculate the major loop, half-way done interrupt point.

The following table summarizes how a DMA channel can link to another DMA channel, i.e, use another channel's TCD, at the end of a loop.

Table 16-1074. Channel Linking Parameters

Desired Link Behavior	TCD Control Field Name	Description
Link at end of Minor Loop	CITER[E_LINK]	Enable channel-to-channel linking on minor loop completion (current iteration)
	CITER[LINKCH]	Link channel number when linking at end of minor loop (current iteration)
Link at end of Major Loop	CSR[MAJOR_E_LINK]	Enable channel-to-channel linking on major loop completion
	CSR[MAJOR_LINKCH]	Link channel number when linking at end of major loop

16.6.7 Dynamic programming

This section provides recommended methods to change the programming model during channel execution.

16.6.7.1 Dynamically changing the channel priority

The following two options are recommended for dynamically changing channel priority levels:

1. Switch to Round-Robin Channel Arbitration mode, change the channel priorities, then switch back to Fixed Arbitration mode,
2. Disable all the channels within a group, change the channel priorities within that group only, then enable the appropriate channels.

The following two options are available for dynamically changing group priority levels:

1. Switch to Round-Robin Group Arbitration mode, change the group priorities, then switch back to Fixed Arbitration mode,
2. Disable all channels, change the group priorities, then enable the appropriate channels.

16.6.7.2 Dynamic channel linking

Dynamic channel linking is the process of setting the TCD.major.e_link bit during channel execution (see the diagram in [TCD structure](#)). This bit is read from the TCD local memory at the end of channel execution, thus allowing the user to enable the feature during channel execution.

Because the user is allowed to change the configuration during execution, a coherency model is needed. Consider the scenario where the user attempts to execute a dynamic channel link by enabling the TCD.major.e_link bit at the same time the eDMA engine is retiring the channel. The TCD.major.e_link would be set in the programmer's model, but it would be unclear whether the actual link was made before the channel retired.

The following coherency model is recommended when executing a dynamic channel link request.

1. Write 1 to the TCD.major.e_link bit.
2. Read back the TCD.major.e_link bit.
3. Test the TCD.major.e_link request status:
 - If TCD.major.e_link = 1, the dynamic link attempt was successful.
 - If TCD.major.e_link = 0, the attempted dynamic link did not succeed (the channel was already retiring).

For this request, the TCD local memory controller forces the TCD.major.e_link bit to zero on any writes to a channel's TCD.word7 after that channel's TCD.done bit is set, indicating the major loop is complete.

NOTE

The user must clear the TCD.done bit before writing the TCD.major.e_link bit. The TCD.done bit is cleared automatically by the eDMA engine after a channel begins execution.

16.6.7.3 Dynamic scatter/gather

Scatter/gather is the process of automatically loading a new TCD into a channel. It allows a DMA channel to use multiple TCDs; this enables a DMA channel to scatter the DMA data to multiple destinations or gather it from multiple sources. When scatter/gather is enabled and the channel has finished its major loop, a new TCD is fetched from system memory and loaded into that channel's descriptor location in eDMA programmer's model, thus replacing the current descriptor.

Because the user is allowed to change the configuration during execution, a coherency model is needed. Consider the scenario where the user attempts to execute a dynamic scatter/gather operation by enabling the TCD.e_sg bit at the same time the eDMA engine is retiring the channel. The TCD.e_sg would be set in the programmer's model, but it would be unclear whether the actual scatter/gather request was honored before the channel retired.

Two methods for this coherency model are shown in the following subsections. Method 1 has the advantage of reading the major.linkch field and the e_sg bit with a single read. For both dynamic channel linking and scatter/gather requests, the TCD local memory controller forces the TCD.major.e_link and TCD.e_sg bits to zero on any writes to a channel's TCD.word7 if that channel's TCD.done bit is set indicating the major loop is complete.

NOTE

The user must clear the TCD.done bit before writing the TCD.major.e_link or TCD.e_sg bits. The TCD.done bit is cleared automatically by the eDMA engine after a channel begins execution.

16.6.7.3.1 Method 1 (channel not using major loop channel linking)

For a channel not using major loop channel linking, the coherency model described here may be used for a dynamic scatter/gather request.

When the TCD.major.e_link bit is zero, the TCD.major.linkch field is not used by the eDMA. In this case, the TCD.major.linkch bits may be used for other purposes. This method uses the TCD.major.linkch field as a TCD identification (ID).

1. When the descriptors are built, write a unique TCD ID in the TCD.major.linkch field for each TCD associated with a channel using dynamic scatter/gather.
2. Write 1b to the TCD.d_req bit.

Should a dynamic scatter/gather attempt fail, setting the TCD.d_req bit will prevent a future hardware activation of this channel. This stops the channel from executing with a destination address (daddr) that was calculated using a scatter/gather address (written in the next step) instead of a dlast final offset value.

3. Write the TCD.dlast_sga field with the scatter/gather address.
4. Write 1b to the TCD.e_sg bit.
5. Read back the 16 bit TCD control/status field.

6. Test the TCD.e_sg request status and TCD.major.linkch value:

If e_sg = 1b, the dynamic link attempt was successful.

If e_sg = 0b and the major.linkch (ID) did not change, the attempted dynamic link did not succeed (the channel was already retiring).

If e_sg = 0b and the major.linkch (ID) changed, the dynamic link attempt was successful (the new TCD's e_sg value cleared the e_sg bit).

16.6.7.3.2 Method 2 (channel using major loop channel linking)

For a channel using major loop channel linking, the coherency model described here may be used for a dynamic scatter/gather request. This method uses the TCD.dlast_sga field as a TCD identification (ID).

1. Write 1b to the TCD.d_req bit.

Should a dynamic scatter/gather attempt fail, setting the d_req bit will prevent a future hardware activation of this channel. This stops the channel from executing with a destination address (daddr) that was calculated using a scatter/gather address (written in the next step) instead of a dlast final offset value.

2. Write the TCD.dlast_sga field with the scatter/gather address.

3. Write 1b to the TCD.e_sg bit.

4. Read back the TCD.e_sg bit.

5. Test the TCD.e_sg request status:

If e_sg = 1b, the dynamic link attempt was successful.

If e_sg = 0b, read the 32 bit TCD dlast_sga field.

If e_sg = 0b and the dlast_sga did not change, the attempted dynamic link did not succeed (the channel was already retiring).

If e_sg = 0b and the dlast_sga changed, the dynamic link attempt was successful (the new TCD's e_sg value cleared the e_sg bit).

Chapter 17

Error Injection Module (EIM)

17.1 Chip-specific EIM information

This chip has one instance of the EIM.

17.1.1 EIM channels

The EIM on this chip supports 6 channels. The number of each channel corresponds to:

- A field in the EIM's EICHEN register
- An EICHD n _Word0 register
- An EICHD n _Word1 register
- An EICHD n _Word2 register

17.1.2 EIM channel assignments

Each EIM channel corresponds to a specific RAM array target. The EIM channel can inject errors on accesses to that specific RAM array. The following table shows the channel assignments.

Table 17-1. RAM array targets of EIM channels

EIM channel	RAM array target
0	PRAMC_0
1	PRAMC_1
2	FEC MIB
3	FEC RIF
4	eDMA_A TCD
5	eDMA_B TCD

17.1.3 EIM_EICHDN_Word register bit mapping

The following table shows the implemented bits of the [Error Injection Channel Descriptor, Word0 \(EIM_EICHDN_WORD0\)](#), [Error Injection Channel Descriptor, Word1 \(EIM_EICHDN_WORD1\)](#), and [Error Injection Channel Descriptor, Word2 \(EIM_EICHDN_WORD2\)](#) registers for each channel.

Off set	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
100	PRAMC_0 CHKBIT_MASK									Reserved																						
104	PRAMC_0 B0_3DATA_MASK (bits 63-32)																															
108	PRAMC_0 B4_7DATA_MASK (bits 31-0)																															
200	PRAMC_1 CHKBIT_MASK									Reserved																						
204	PRAMC_1 B0_3DATA_MASK (bits 63-32)																															
208	PRAMC_1 B4_7DATA_MASK (bits 31-0)																															
300	FEC MIB CHKBIT_MASK									Reserved																						
304	FEC MIB B0_3DATA_MASK (bits 31-0)																															
308	Reserved																															
400	FEC RIF CHKBIT_MASK									Reserved																						
404	Reserved																											FEC RIF B0_3DATA_MASK (bits 35-32)				
408	FEC RIF B4_7DATA_MASK (bits 31-0)																															
500	eDMA_A TCD CHKBIT_MASK									Reserved																						
504	eDMA_A TCD B0_3DATA_MASK (bits 63-32)																															
508	eDMA_A TCD B4_7DATA_MASK (bits 31-0)																															
600	eDMA_B TCD CHKBIT_MASK									Reserved																						
604	eDMA_B TCD B0_3DATA_MASK (bits 63-32)																															
608	eDMA_B TCD B4_7DATA_MASK (bits 31-0)																															

17.2 Introduction

The Error Injection module is mainly used for diagnostic purposes. It provides a method for diagnostic coverage of the peripheral memories. See the chip-specific EIM information to determine which peripheral memories are supported by this method.

17.2.1 Overview

The Error Injection Module (EIM) provides support for inducing single-bit and multi-bit inversions on read data when accessing peripheral RAMs. Injecting faults on memory accesses can be used to exercise the SEC-DED ECC function of the related system.

NOTE

The following diagram shows an example EIM implementation with a 64-bit read data bus and an 8-bit checkbit bus.

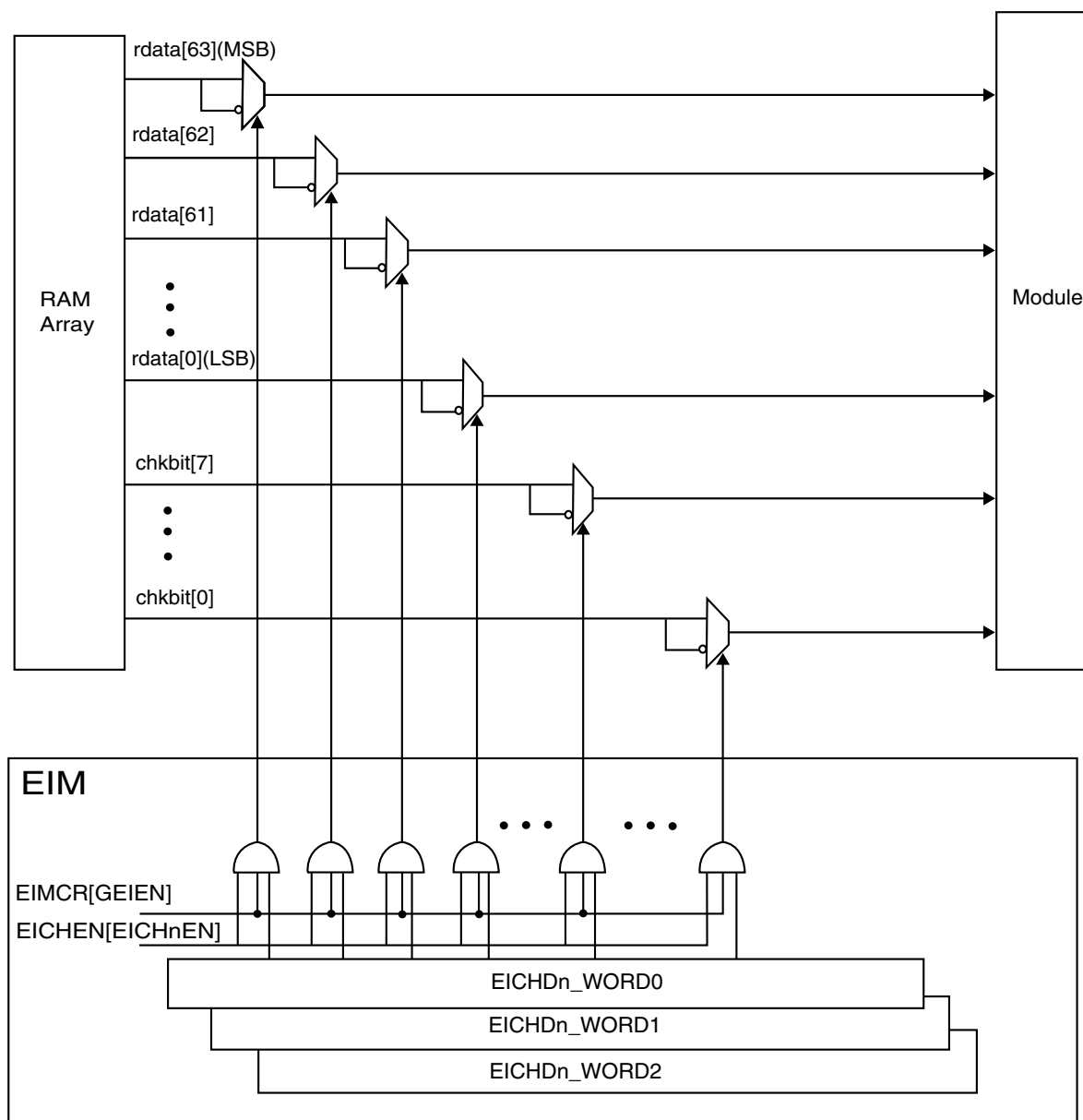


Figure 17-1. EIM functional block diagram (64-bit read data bus and 8-bit checkbit bus)

17.2.2 Features

The EIM includes these features:

- Supports 6 error injection channels
- Protection against accidental enable and reconfiguration error injection function via two-stage enable mechanism

17.3 Memory map and register definition

The EIM provides an IPS programming model mapped to an on-platform peripheral slot.

Programming model access

All system bus masters can access the programming model:

- Only in supervisor mode
- Using only 32-bit (word) accesses

Any of the following attempted references to the programming model generates an IPS error termination:

- In user mode
- Using non-32-bit access sizes
- To undefined (reserved) addresses

Attempted updates to the programming model while the EIM is in the midst of an operation will result in non-deterministic behavior.

Error injection channel descriptor: function and structure

Each error injection channel descriptor:

- Specifies a mask that defines which bits of the read data and checkbit bus from target RAM are inverted on a read access.
- Consists of a 128-bit (16-byte) structure, composed of four 32-bit words, in the EIM programming model.

- Each descriptor consists of Error Injection Channel Descriptor Word0-1 (EICHD n _WORD0-1) and, depending on the implementation, Word2 (EICHD n _WORD2).
- When Word2 is unused, the corresponding word in the 16-byte structure is unused.
- The fourth word in the 16-byte structure is always unused.

The multiple channel descriptors are organized sequentially.

EIM memory map

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
0	Error Injection Module Configuration Register (EIM_EIMCR)	32	R/W	0000_0000h	17.3.1/756
4	Error Injection Channel Enable register (EIM_EICHEN)	32	R/W	0000_0000h	17.3.2/757
100	Error Injection Channel Descriptor, Word0 (EIM_EICHD0_WORD0)	32	R/W	0000_0000h	17.3.3/759
104	Error Injection Channel Descriptor, Word1 (EIM_EICHD0_WORD1)	32	R/W	0000_0000h	17.3.4/760
108	Error Injection Channel Descriptor, Word2 (EIM_EICHD0_WORD2)	32	R/W	0000_0000h	17.3.5/760
200	Error Injection Channel Descriptor, Word0 (EIM_EICHD1_WORD0)	32	R/W	0000_0000h	17.3.3/759
204	Error Injection Channel Descriptor, Word1 (EIM_EICHD1_WORD1)	32	R/W	0000_0000h	17.3.4/760
208	Error Injection Channel Descriptor, Word2 (EIM_EICHD1_WORD2)	32	R/W	0000_0000h	17.3.5/760
300	Error Injection Channel Descriptor, Word0 (EIM_EICHD2_WORD0)	32	R/W	0000_0000h	17.3.3/759
304	Error Injection Channel Descriptor, Word1 (EIM_EICHD2_WORD1)	32	R/W	0000_0000h	17.3.4/760
308	Error Injection Channel Descriptor, Word2 (EIM_EICHD2_WORD2)	32	R/W	0000_0000h	17.3.5/760
400	Error Injection Channel Descriptor, Word0 (EIM_EICHD3_WORD0)	32	R/W	0000_0000h	17.3.3/759
404	Error Injection Channel Descriptor, Word1 (EIM_EICHD3_WORD1)	32	R/W	0000_0000h	17.3.4/760
408	Error Injection Channel Descriptor, Word2 (EIM_EICHD3_WORD2)	32	R/W	0000_0000h	17.3.5/760
500	Error Injection Channel Descriptor, Word0 (EIM_EICHD4_WORD0)	32	R/W	0000_0000h	17.3.3/759
504	Error Injection Channel Descriptor, Word1 (EIM_EICHD4_WORD1)	32	R/W	0000_0000h	17.3.4/760
508	Error Injection Channel Descriptor, Word2 (EIM_EICHD4_WORD2)	32	R/W	0000_0000h	17.3.5/760

Table continues on the next page...

EIM memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
600	Error Injection Channel Descriptor, Word0 (EIM_EICHD5_WORD0)	32	R/W	0000_0000h	17.3.3/759
604	Error Injection Channel Descriptor, Word1 (EIM_EICHD5_WORD1)	32	R/W	0000_0000h	17.3.4/760
608	Error Injection Channel Descriptor, Word2 (EIM_EICHD5_WORD2)	32	R/W	0000_0000h	17.3.5/760

17.3.1 Error Injection Module Configuration Register (EIM_EIMCR)

The EIM Configuration Register is used to globally enable/disable the error injection function.

Address: 0h base + 0h offset = 0h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15															
R	0																														
W																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0															
Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31															
R	0																														GEIEN
W																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0															

EIM_EIMCR field descriptions

Field	Description
0–30 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
31 GEIEN	Global Error Injection Enable This bit globally enables or disables the error injection function of the EIM. This field is initialized by hardware reset. 0 Disabled 1 Enabled

17.3.2 Error Injection Channel Enable register (EIM_EICHEN)

Each field of the Error Injection Channel Enable register (EICHEN) is used to enable or disable the corresponding error injection channel.

NOTE

To enable an error injection channel, the Global Error Injection Enable (EIMCR[GEIEN]) field must also be asserted.

Address: 0h base + 4h offset = 4h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	EICH0EN	EICH1EN	EICH2EN	EICH3EN	EICH4EN	EICH5EN	0									
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

EIM_EICHEN field descriptions

Field	Description
0 EICH0EN	Error Injection Channel 0 Enable This field enables the corresponding error injection channel. The Global Error Injection Enable (EIMCR[GEIEN]) field must also be asserted to enable error injection. After error injection is enabled, all subsequent read accesses incur one or more bit inversions as defined in the corresponding EICHn_WORD registers. Error injection remains in effect until the error injection channel is manually disabled via software. Any write to the corresponding EICHn_WORD registers clears the corresponding EICHEN[EICHnEN] field, disabling the error injection channel. 0 Error injection is disabled on Error Injection Channel 0 1 Error injection is enabled on Error Injection Channel 0
1 EICH1EN	Error Injection Channel 1 Enable This field enables the corresponding error injection channel. The Global Error Injection Enable (EIMCR[GEIEN]) field must also be asserted to enable error injection. After error injection is enabled, all subsequent read accesses incur one or more bit inversions as defined in the corresponding EICHn_WORD registers. Error injection remains in effect until the error injection channel is manually disabled via software. Any write to the corresponding EICHn_WORD registers clears the corresponding EICHEN[EICHnEN] field, disabling the error injection channel.

Table continues on the next page...

EIM_EICHEN field descriptions (continued)

Field	Description
	0 Error injection is disabled on Error Injection Channel 1 1 Error injection is enabled on Error Injection Channel 1
2 EICH2EN	Error Injection Channel 2 Enable This field enables the corresponding error injection channel. The Global Error Injection Enable (EIMCR[GEIEN]) field must also be asserted to enable error injection. After error injection is enabled, all subsequent read accesses incur one or more bit inversions as defined in the corresponding EICHD n _WORD registers. Error injection remains in effect until the error injection channel is manually disabled via software. Any write to the corresponding EICHD n _WORD registers clears the corresponding EICHEN[EICH n EN] field, disabling the error injection channel. 0 Error injection is disabled on Error Injection Channel 2 1 Error injection is enabled on Error Injection Channel 2
3 EICH3EN	Error Injection Channel 3 Enable This field enables the corresponding error injection channel. The Global Error Injection Enable (EIMCR[GEIEN]) field must also be asserted to enable error injection. After error injection is enabled, all subsequent read accesses incur one or more bit inversions as defined in the corresponding EICHD n _WORD registers. Error injection remains in effect until the error injection channel is manually disabled via software. Any write to the corresponding EICHD n _WORD registers clears the corresponding EICHEN[EICH n EN] field, disabling the error injection channel. 0 Error injection is disabled on Error Injection Channel 3 1 Error injection is enabled on Error Injection Channel 3
4 EICH4EN	Error Injection Channel 4 Enable This field enables the corresponding error injection channel. The Global Error Injection Enable (EIMCR[GEIEN]) field must also be asserted to enable error injection. After error injection is enabled, all subsequent read accesses incur one or more bit inversions as defined in the corresponding EICHD n _WORD registers. Error injection remains in effect until the error injection channel is manually disabled via software. Any write to the corresponding EICHD n _WORD registers clears the corresponding EICHEN[EICH n EN] field, disabling the error injection channel. 0 Error injection is disabled on Error Injection Channel 4 1 Error injection is enabled on Error Injection Channel 4
5 EICH5EN	Error Injection Channel 5 Enable This field enables the corresponding error injection channel. The Global Error Injection Enable (EIMCR[GEIEN]) field must also be asserted to enable error injection. After error injection is enabled, all subsequent read accesses incur one or more bit inversions as defined in the corresponding EICHD n _WORD registers. Error injection remains in effect until the error injection channel is manually disabled via software. Any write to the corresponding EICHD n _WORD registers clears the corresponding EICHEN[EICH n EN] field, disabling the error injection channel. 0 Error injection is disabled on Error Injection Channel 5 1 Error injection is enabled on Error Injection Channel 5

Table continues on the next page...

EIM_EICHEN field descriptions (continued)

Field	Description
6–31 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

**17.3.3 Error Injection Channel Descriptor, Word0
(EIM_EICHD_n_WORD0)**

The first word of the Error Injection Channel Descriptor defines a left-justified mask field (CHKBIT_MASK) whose width is up to 8 bits. Each bit of CHKBIT_MASK specifies whether the corresponding bit of the checkbit bus from the target RAM should be inverted or remain unmodified on read accesses. Successful writes to this field clear the corresponding error injection channel valid bit, EICHEN[EICH_nEN].

Address: 0h base + 100h offset + (256d × i), where i=0d to 5d

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	CHKBIT_MASK								0																							
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	

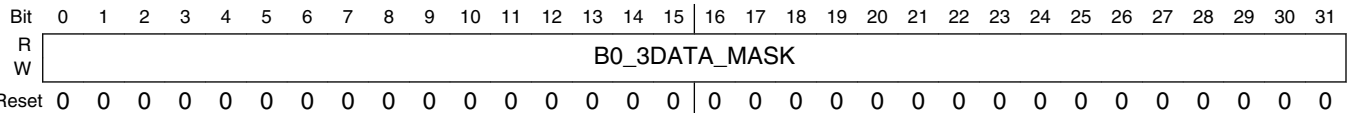
EIM_EICHD_n_WORD0 field descriptions

Field	Description
0–7 CHKBIT_MASK	Checkbit Mask This field defines a bit-mapped mask that specifies whether the corresponding bit of the checkbit bus from the target RAM should be inverted or remain unmodified. The width of the field for the channel is defined in the chip-specific EIM information. 0 The corresponding bit of the checkbit bus remains unmodified. 1 The corresponding bit of the checkbit bus is inverted.
8–31 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

17.3.4 Error Injection Channel Descriptor, Word1
(EIM_EICHDn_WORD1)

The second word of the Error Injection Channel Descriptor defines the bits in the mask corresponding to bytes 0–3 of the read data bus. Each bit specifies whether the corresponding bit of the read data bus from the target RAM should be inverted or remain unmodified on read accesses. Successful writes to this field clear the corresponding error injection channel valid field, EICHEN[EICHnEN].

Address: 0h base + 104h offset + (256d × i), where i=0d to 5d



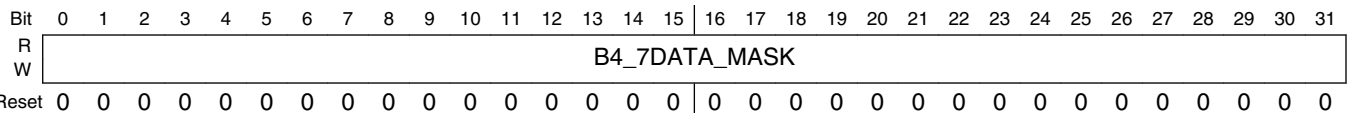
EIM_EICHDn_WORD1 field descriptions

Field	Description
0–31 B0_3DATA_MASK	Data Mask Bytes 0-3 This field defines a bit-mapped mask that specifies whether the corresponding bit of the read data bus from the target RAM should be inverted or remain unmodified. For details about which bits in this field are implemented for the applicable channel and how the bits map to bytes 0-3 of the read data bus, see the chip-specific EIM information. 0 The corresponding bit of bytes 0-3 on the read data bus remains unmodified 1 The corresponding bit of bytes 0-3 on the read data bus is inverted.

17.3.5 Error Injection Channel Descriptor, Word2
(EIM_EICHDn_WORD2)

The third word of the Error Injection Channel Descriptor, when present, defines the bits in the mask corresponding to bytes 4–7 of the read data bus. Each bit specifies whether the corresponding bit of the read data bus from the target RAM should be inverted or remain unmodified on read accesses. Successful writes to this field clear the corresponding error injection channel valid field, EICHEN[EICHnEN].

Address: 0h base + 108h offset + (256d × i), where i=0d to 5d



EIM_EICHD n _WORD2 field descriptions

Field	Description
0–31 B4_7DATA_ MASK	Data Mask Bytes 4-7 This field defines a bit-mapped mask that specifies whether the corresponding bit of the read data bus from the target RAM should be inverted or remain unmodified. For details about which bits in this field are implemented for the applicable channel and how the bits map to bytes 4-7 of the read data bus, see the chip-specific EIM information. 0 The corresponding bit of bytes 4-7 on the read data bus remains unmodified. 1 The corresponding bit of bytes 4-7 on the read data bus is inverted.

17.4 Functional description

The EIM provides protection against accidental enabling and reconfiguration of the error injection function by enforcing a two-stage enablement mechanism. To properly enable the error injection mechanism for a channel:

- Write 1 to the EICHEN[EICH n EN] field, where n denotes the channel number.
- Write 1 to EIMCR[GEIEN].

NOTE

When the use case for a channel requires writing any EICHD n _WORD register, write the EICHD n _WORD register before executing the two-stage enablement mechanism. A successful write to any EICHD n _WORD register clears the corresponding EICHEN[EICH n EN] field.

The EIM supports 6 error injection channels. Each channel:

- is assigned to a single memory array interface.
- intercepts the assigned memory read data bus and checkbit bus and injects errors by inverting the value transmitted for selected bits on each bus line.

On a memory read access, the applicable EICHD n _WORD registers define which bits of the read data and/or checkbit bus to invert.

Figure 17-1 depicts the interception and override of a 64-bit read data bus and an 8-bit checkbit data bus for an example memory array.

Chapter 18

Error Reporting Module (ERM)

18.1 Chip-specific ERM information

This chip has one instance of the ERM.

18.1.1 ERM memory channels

The ERM on this chip supports 20 memory channels. The number of each memory channel corresponds to:

- A field in the ERM's CR0, CR1, or CR2 register
- A field in the ERM's SR0, SR1, or SR2 register
- An ERM_EAR_n register
- An ERM_SYN_n register

18.1.2 Sources of memory error events

Each ERM channel corresponds to a source of potential memory error events. The following table shows the channel assignments.

Table 18-1. Memory error event sources for ERM channels

ERM channel	Memory event source
0	PRAMC_0
1	PRAMC_1
2	eDMA_A TCD ¹
3	eDMA_B TCD ¹
4	FEC MIB ¹

Table continues on the next page...

Table 18-1. Memory error event sources for ERM channels (continued)

ERM channel	Memory event source
5	FEC RIF ¹
6	PFLASH port 0
7	PFLASH port 1
8	AIPS_0
9	AIPS_1
10	FEC e2eECC
11	CSE
12	SIPI
13	Core0 instruction
14	Core0 data
15	Core1 instruction
16	Core1 data
17	eDMA_A e2eECC
18	eDMA_B e2eECC
19	EBI e2eECC

1. For these memory elements, the error address captured is the relative array address and not the system address.

18.1.3 Managing single-bit faults of TCD RAM for eDMA_A

When a single-bit ECC event occurs during a read of the TCD RAM for eDMA_A, the fault is correctly identified and forwarded to the ERM. However, the fault is not corrected before being used by eDMA_A.

To prevent eDMA_A's use of single-bit faults from the TCD RAM for eDMA_A: Ensure that ERM_CR0[ESCIE2] is set to 1 to enable the ERM to trigger an interrupt, and use the interrupt to manage properly any single-bit fault that occurs.

18.2 Introduction

18.2.1 Overview

The Error Reporting Module (ERM) provides information and optional interrupt notification on memory error events associated with ECC (Error Correction Code). The ERM collects ECC events on memory accesses for platform local memory arrays, such as

flash memory, system RAM, or peripheral RAMs. See the chip-specific ERM information for details about supported memory sources and specific memory channel assignments.

18.2.2 Features

The ERM includes these features:

- Capture address and syndrome information on single-bit correction and non-correctable ECC events
- Optional interrupt notification on captured ECC events
- Support for ECC event capturing for memory sources, with individual reporting fields and interrupt configuration per memory channel.

18.3 Memory map and register definition

The ERM provides an IPS programming model mapped to an on-platform peripheral slot. The programming model can only be referenced using a 32-bit (word) access. Attempted references using different access sizes or to undefined (reserved) addresses or in user mode generates an IPS error termination. The ERM allows access to the programming model by all system bus masters.

The programming model can only be accessed in supervisor mode.

Attempted updates to the programming model while the ERM module is in the middle of an operation will result in non-deterministic behavior.

ERM memory map

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
0	ERM Configuration Register (ERM_CR0)	32	R/W	0000_0000h	18.3.1/767
4	ERM Configuration Register (ERM_CR1)	32	R/W	0000_0000h	18.3.2/770
8	ERM Configuration Register (ERM_CR2)	32	R/W	0000_0000h	18.3.3/773
10	ERM Status Register (ERM_SR0)	32	w1c	0000_0000h	18.3.4/776
14	ERM Status Register (ERM_SR1)	32	w1c	0000_0000h	18.3.5/780
18	ERM Status Register (ERM_SR2)	32	w1c	0000_0000h	18.3.6/784
100	ERM Memory n Error Address Register (ERM_EAR0)	32	R	0000_0000h	18.3.7/786
104	ERM Memory n Syndrome Register (ERM_SYNO)	32	R	0000_0000h	18.3.8/787
110	ERM Memory n Error Address Register (ERM_EAR1)	32	R	0000_0000h	18.3.7/786

Table continues on the next page...

ERM memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
114	ERM Memory n Syndrome Register (ERM_SYN1)	32	R	0000_0000h	18.3.8/787
120	ERM Memory n Error Address Register (ERM_EAR2)	32	R	0000_0000h	18.3.7/786
124	ERM Memory n Syndrome Register (ERM_SYN2)	32	R	0000_0000h	18.3.8/787
130	ERM Memory n Error Address Register (ERM_EAR3)	32	R	0000_0000h	18.3.7/786
134	ERM Memory n Syndrome Register (ERM_SYN3)	32	R	0000_0000h	18.3.8/787
140	ERM Memory n Error Address Register (ERM_EAR4)	32	R	0000_0000h	18.3.7/786
144	ERM Memory n Syndrome Register (ERM_SYN4)	32	R	0000_0000h	18.3.8/787
150	ERM Memory n Error Address Register (ERM_EAR5)	32	R	0000_0000h	18.3.7/786
154	ERM Memory n Syndrome Register (ERM_SYN5)	32	R	0000_0000h	18.3.8/787
160	ERM Memory n Error Address Register (ERM_EAR6)	32	R	0000_0000h	18.3.7/786
164	ERM Memory n Syndrome Register (ERM_SYN6)	32	R	0000_0000h	18.3.8/787
170	ERM Memory n Error Address Register (ERM_EAR7)	32	R	0000_0000h	18.3.7/786
174	ERM Memory n Syndrome Register (ERM_SYN7)	32	R	0000_0000h	18.3.8/787
180	ERM Memory n Error Address Register (ERM_EAR8)	32	R	0000_0000h	18.3.7/786
184	ERM Memory n Syndrome Register (ERM_SYN8)	32	R	0000_0000h	18.3.8/787
190	ERM Memory n Error Address Register (ERM_EAR9)	32	R	0000_0000h	18.3.7/786
194	ERM Memory n Syndrome Register (ERM_SYN9)	32	R	0000_0000h	18.3.8/787
1A0	ERM Memory n Error Address Register (ERM_EAR10)	32	R	0000_0000h	18.3.7/786
1A4	ERM Memory n Syndrome Register (ERM_SYN10)	32	R	0000_0000h	18.3.8/787
1B0	ERM Memory n Error Address Register (ERM_EAR11)	32	R	0000_0000h	18.3.7/786
1B4	ERM Memory n Syndrome Register (ERM_SYN11)	32	R	0000_0000h	18.3.8/787
1C0	ERM Memory n Error Address Register (ERM_EAR12)	32	R	0000_0000h	18.3.7/786
1C4	ERM Memory n Syndrome Register (ERM_SYN12)	32	R	0000_0000h	18.3.8/787
1D0	ERM Memory n Error Address Register (ERM_EAR13)	32	R	0000_0000h	18.3.7/786
1D4	ERM Memory n Syndrome Register (ERM_SYN13)	32	R	0000_0000h	18.3.8/787
1E0	ERM Memory n Error Address Register (ERM_EAR14)	32	R	0000_0000h	18.3.7/786
1E4	ERM Memory n Syndrome Register (ERM_SYN14)	32	R	0000_0000h	18.3.8/787
1F0	ERM Memory n Error Address Register (ERM_EAR15)	32	R	0000_0000h	18.3.7/786
1F4	ERM Memory n Syndrome Register (ERM_SYN15)	32	R	0000_0000h	18.3.8/787
200	ERM Memory n Error Address Register (ERM_EAR16)	32	R	0000_0000h	18.3.7/786
204	ERM Memory n Syndrome Register (ERM_SYN16)	32	R	0000_0000h	18.3.8/787
210	ERM Memory n Error Address Register (ERM_EAR17)	32	R	0000_0000h	18.3.7/786
214	ERM Memory n Syndrome Register (ERM_SYN17)	32	R	0000_0000h	18.3.8/787
220	ERM Memory n Error Address Register (ERM_EAR18)	32	R	0000_0000h	18.3.7/786
224	ERM Memory n Syndrome Register (ERM_SYN18)	32	R	0000_0000h	18.3.8/787
230	ERM Memory n Error Address Register (ERM_EAR19)	32	R	0000_0000h	18.3.7/786
234	ERM Memory n Syndrome Register (ERM_SYN19)	32	R	0000_0000h	18.3.8/787

18.3.1 ERM Configuration Register (ERM_CR0)

The ERM Configuration Register is a 32-bit control register for configuring the interrupt notification capability for memory channels 0 to 7.

Address: 0h base + 0h offset = 0h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	ESCIE0	ENCIE0	0		ESCIE1	ENCIE1	0		ESCIE2	ENCIE2	0		ESCIE3	ENCIE3	0	
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	ESCIE4	ENCIE4	0		ESCIE5	ENCIE5	0		ESCIE6	ENCIE6	0		ESCIE7	ENCIE7	0	
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

ERM_CR0 field descriptions

Field	Description
0 ESCIE0	Enable Memory 0 Single Correction Interrupt Notification This field is initialized by hardware reset. NOTE: Refer to the chip-specific ERM information for details on Memory 0 mapping. 0 Interrupt notification of Memory 0 single-bit correction events is disabled. 1 Interrupt notification of Memory 0 single-bit correction events is enabled.
1 ENCIE0	Enable Memory 0 Non-Correctable Interrupt Notification This field is initialized by hardware reset. NOTE: Refer to the chip-specific ERM information for details on Memory 0 mapping. 0 Interrupt notification of Memory 0 non-correctable error events is disabled. 1 Interrupt notification of Memory 0 non-correctable error events is enabled.
2–3 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
4 ESCIE1	Enable Memory 1 Single Correction Interrupt Notification This field is initialized by hardware reset. NOTE: Refer to the chip-specific ERM information for details on Memory 1 mapping. 0 Interrupt notification of Memory 1 single-bit correction events is disabled. 1 Interrupt notification of Memory 1 single-bit correction events is enabled.

Table continues on the next page...

ERM_CR0 field descriptions (continued)

Field	Description
5 ENCIE1	Enable Memory 1 Non-Correctable Interrupt Notification This field is initialized by hardware reset. NOTE: Refer to the chip-specific ERM information for details on Memory 1 mapping. 0 Interrupt notification of Memory 1 non-correctable error events is disabled. 1 Interrupt notification of Memory 1 non-correctable error events is enabled.
6–7 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
8 ESCIE2	Enable Memory 2 Single Correction Interrupt Notification This field is initialized by hardware reset. NOTE: Refer to the chip-specific ERM information for details on Memory 2 mapping. 0 Interrupt notification of Memory 2 single-bit correction events is disabled. 1 Interrupt notification of Memory 2 single-bit correction events is enabled.
9 ENCIE2	Enable Memory 2 Non-Correctable Interrupt Notification This field is initialized by hardware reset. NOTE: Refer to the chip-specific ERM information for details on Memory 2 mapping. 0 Interrupt notification of Memory 2 non-correctable error events is disabled. 1 Interrupt notification of Memory 2 non-correctable error events is enabled.
10–11 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
12 ESCIE3	Enable Memory 3 Single Correction Interrupt Notification This field is initialized by hardware reset. NOTE: Refer to the chip-specific ERM information for details on Memory 3 mapping. 0 Interrupt notification of Memory 3 single-bit correction events is disabled. 1 Interrupt notification of Memory 3 single-bit correction events is enabled.
13 ENCIE3	Enable Memory 3 Non-Correctable Interrupt Notification This field is initialized by hardware reset. NOTE: Refer to the chip-specific ERM information for details on Memory 3 mapping. 0 Interrupt notification of Memory 3 non-correctable error events is disabled. 1 Interrupt notification of Memory 3 non-correctable error events is enabled.
14–15 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
16 ESCIE4	Enable Memory 4 Single Correction Interrupt Notification This field is initialized by hardware reset. NOTE: Refer to the chip-specific ERM information for details on Memory 4 mapping. 0 Interrupt notification of Memory 4 single-bit correction events is disabled. 1 Interrupt notification of Memory 4 single-bit correction events is enabled.
17 ENCIE4	Enable Memory 4 Non-Correctable Interrupt Notification

Table continues on the next page...

ERM_CR0 field descriptions (continued)

Field	Description
	This field is initialized by hardware reset. NOTE: Refer to the chip-specific ERM information for details on Memory 4 mapping. 0 Interrupt notification of Memory 4 non-correctable error events is disabled. 1 Interrupt notification of Memory 4 non-correctable error events is enabled.
18–19 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
20 ESCIE5	Enable Memory 5 Single Correction Interrupt Notification This field is initialized by hardware reset. NOTE: Refer to the chip-specific ERM information for details on Memory 5 mapping. 0 Interrupt notification of Memory 5 single-bit correction events is disabled. 1 Interrupt notification of Memory 5 single-bit correction events is enabled.
21 ENCIE5	Enable Memory 5 Non-Correctable Interrupt Notification This field is initialized by hardware reset. NOTE: Refer to the chip-specific ERM information for details on Memory 5 mapping. 0 Interrupt notification of Memory 5 non-correctable error events is disabled. 1 Interrupt notification of Memory 5 non-correctable error events is enabled.
22–23 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
24 ESCIE6	Enable Memory 6 Single Correction Interrupt Notification This field is initialized by hardware reset. NOTE: Refer to the chip-specific ERM information for details on Memory 6 mapping. 0 Interrupt notification of Memory 6 single-bit correction events is disabled. 1 Interrupt notification of Memory 6 single-bit correction events is enabled.
25 ENCIE6	Enable Memory 6 Non-Correctable Interrupt Notification This field is initialized by hardware reset. NOTE: Refer to the chip-specific ERM information for details on Memory 6 mapping. 0 Interrupt notification of Memory 6 non-correctable error events is disabled. 1 Interrupt notification of Memory 6 non-correctable error events is enabled.
26–27 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
28 ESCIE7	Enable Memory 7 Single Correction Interrupt Notification This field is initialized by hardware reset. NOTE: Refer to the chip-specific ERM information for details on Memory 7 mapping. 0 Interrupt notification of Memory 7 single-bit correction events is disabled. 1 Interrupt notification of Memory 7 single-bit correction events is enabled.
29 ENCIE7	Enable Memory 7 Non-Correctable Interrupt Notification This field is initialized by hardware reset.

Table continues on the next page...

ERM_CR0 field descriptions (continued)

Field	Description
	NOTE: Refer to the chip-specific ERM information for details on Memory 7 mapping. 0 Interrupt notification of Memory 7 non-correctable error events is disabled. 1 Interrupt notification of Memory 7 non-correctable error events is enabled.
30–31 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

18.3.2 ERM Configuration Register (ERM_CR1)

The ERM Configuration Register is a 32-bit control register for configuring the interrupt notification capability for memory channels 8 to 15.

Address: 0h base + 4h offset = 4h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	ESCIE8	ENCIE8	0		ESCIE9	ENCIE9	0		ESCIE10	ENCIE10	0		ESCIE11	ENCIE11	0	
W	ESCIE8	ENCIE8			ESCIE9	ENCIE9			ESCIE10	ENCIE10			ESCIE11	ENCIE11		
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	ESCIE12	ENCIE12	0		ESCIE13	ENCIE13	0		ESCIE14	ENCIE14	0		ESCIE15	ENCIE15	0	
W	ESCIE12	ENCIE12			ESCIE13	ENCIE13			ESCIE14	ENCIE14			ESCIE15	ENCIE15		
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

ERM_CR1 field descriptions

Field	Description
0 ESCIE8	Enable Memory 8 Single Correction Interrupt Notification This field is initialized by hardware reset. NOTE: Refer to the chip-specific ERM information for details on Memory 8 mapping. 0 Interrupt notification of Memory 8 single-bit correction events is disabled. 1 Interrupt notification of Memory 8 single-bit correction events is enabled.
1 ENCIE8	Enable Memory 8 Non-Correctable Interrupt Notification This field is initialized by hardware reset. NOTE: Refer to the chip-specific ERM information for details on Memory 8 mapping.

Table continues on the next page...

ERM_CR1 field descriptions (continued)

Field	Description
	0 Interrupt notification of Memory 8 non-correctable error events is disabled. 1 Interrupt notification of Memory 8 non-correctable error events is enabled.
2–3 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
4 ESCIE9	Enable Memory 9 Single Correction Interrupt Notification This field is initialized by hardware reset. NOTE: Refer to the chip-specific ERM information for details on Memory 9 mapping. 0 Interrupt notification of Memory 9 single-bit correction events is disabled. 1 Interrupt notification of Memory 9 single-bit correction events is enabled.
5 ENCIE9	Enable Memory 9 Non-Correctable Interrupt Notification This field is initialized by hardware reset. NOTE: Refer to the chip-specific ERM information for details on Memory 9 mapping. 0 Interrupt notification of Memory 9 non-correctable error events is disabled. 1 Interrupt notification of Memory 9 non-correctable error events is enabled.
6–7 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
8 ESCIE10	Enable Memory 10 Single Correction Interrupt Notification This field is initialized by hardware reset. NOTE: Refer to the chip-specific ERM information for details on Memory 10 mapping. 0 Interrupt notification of Memory 10 single-bit correction events is disabled. 1 Interrupt notification of Memory 10 single-bit correction events is enabled.
9 ENCIE10	Enable Memory 10 Non-Correctable Interrupt Notification This field is initialized by hardware reset. NOTE: Refer to the chip-specific ERM information for details on Memory 10 mapping. 0 Interrupt notification of Memory 10 non-correctable error events is disabled. 1 Interrupt notification of Memory 10 non-correctable error events is enabled.
10–11 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
12 ESCIE11	Enable Memory 11 Single Correction Interrupt Notification This field is initialized by hardware reset. NOTE: Refer to the chip-specific ERM information for details on Memory 11 mapping. 0 Interrupt notification of Memory 11 single-bit correction events is disabled. 1 Interrupt notification of Memory 11 single-bit correction events is enabled.
13 ENCIE11	Enable Memory 11 Non-Correctable Interrupt Notification This field is initialized by hardware reset. NOTE: Refer to the chip-specific ERM information for details on Memory 11 mapping. 0 Interrupt notification of Memory 11 non-correctable error events is disabled. 1 Interrupt notification of Memory 11 non-correctable error events is enabled.

Table continues on the next page...

ERM_CR1 field descriptions (continued)

Field	Description
14–15 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
16 ESCIE12	Enable Memory 12 Single Correction Interrupt Notification This field is initialized by hardware reset. NOTE: Refer to the chip-specific ERM information for details on Memory 12 mapping. 0 Interrupt notification of Memory 12 single-bit correction events is disabled. 1 Interrupt notification of Memory 12 single-bit correction events is enabled.
17 ENCIE12	Enable Memory 12 Non-Correctable Interrupt Notification This field is initialized by hardware reset. NOTE: Refer to the chip-specific ERM information for details on Memory 12 mapping. 0 Interrupt notification of Memory 12 non-correctable error events is disabled. 1 Interrupt notification of Memory 12 non-correctable error events is enabled.
18–19 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
20 ESCIE13	Enable Memory 13 Single Correction Interrupt Notification This field is initialized by hardware reset. NOTE: Refer to the chip-specific ERM information for details on Memory 13 mapping. 0 Interrupt notification of Memory 13 single-bit correction events is disabled. 1 Interrupt notification of Memory 13 single-bit correction events is enabled.
21 ENCIE13	Enable Memory 13 Non-Correctable Interrupt Notification This field is initialized by hardware reset. NOTE: Refer to the chip-specific ERM information for details on Memory 13 mapping. 0 Interrupt notification of Memory 13 non-correctable error events is disabled. 1 Interrupt notification of Memory 13 non-correctable error events is enabled.
22–23 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
24 ESCIE14	Enable Memory 14 Single Correction Interrupt Notification This field is initialized by hardware reset. NOTE: Refer to the chip-specific ERM information for details on Memory 14 mapping. 0 Interrupt notification of Memory 14 single-bit correction events is disabled. 1 Interrupt notification of Memory 14 single-bit correction events is enabled.
25 ENCIE14	Enable Memory 14 Non-Correctable Interrupt Notification This field is initialized by hardware reset. NOTE: Refer to the chip-specific ERM information for details on Memory 14 mapping. 0 Interrupt notification of Memory 14 non-correctable error events is disabled. 1 Interrupt notification of Memory 14 non-correctable error events is enabled.
26–27 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

Table continues on the next page...

ERM_CR1 field descriptions (continued)

Field	Description
28 ESCIE15	Enable Memory 15 Single Correction Interrupt Notification This field is initialized by hardware reset. NOTE: Refer to the chip-specific ERM information for details on Memory 15 mapping. 0 Interrupt notification of Memory 15 single-bit correction events is disabled. 1 Interrupt notification of Memory 15 single-bit correction events is enabled.
29 ENCIE15	Enable Memory 15 Non-Correctable Interrupt Notification This field is initialized by hardware reset. NOTE: Refer to the chip-specific ERM information for details on Memory 15 mapping. 0 Interrupt notification of Memory 15 non-correctable error events is disabled. 1 Interrupt notification of Memory 15 non-correctable error events is enabled.
30–31 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

18.3.3 ERM Configuration Register (ERM_CR2)

The ERM Configuration Register is a 32-bit control register for configuring the interrupt notification capability for memory channels 16 to 19.

Address: 0h base + 8h offset = 8h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	ESCIE16	ENCIE16	0		ESCIE17	ENCIE17	0		ESCIE18	ENCIE18	0		ESCIE19	ENCIE19	0	
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	Reserved	0			Reserved		0		Reserved		0		Reserved		0	
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

ERM_CR2 field descriptions

Field	Description
0 ESCIE16	Enable Memory 16 Single Correction Interrupt Notification This field is initialized by hardware reset. NOTE: Refer to the chip-specific ERM information for details on Memory 16 mapping.

Table continues on the next page...

ERM_CR2 field descriptions (continued)

Field	Description
	0 Interrupt notification of Memory 16 single-bit correction events is disabled. 1 Interrupt notification of Memory 16 single-bit correction events is enabled.
1 ENCIE16	Enable Memory 16 Non-Correctable Interrupt Notification This field is initialized by hardware reset. NOTE: Refer to the chip-specific ERM information for details on Memory 16 mapping. 0 Interrupt notification of Memory 16 non-correctable error events is disabled. 1 Interrupt notification of Memory 16 non-correctable error events is enabled.
2–3 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
4 ESCIE17	Enable Memory 17 Single Correction Interrupt Notification This field is initialized by hardware reset. NOTE: Refer to the chip-specific ERM information for details on Memory 17 mapping. 0 Interrupt notification of Memory 17 single-bit correction events is disabled. 1 Interrupt notification of Memory 17 single-bit correction events is enabled.
5 ENCIE17	Enable Memory 17 Non-Correctable Interrupt Notification This field is initialized by hardware reset. NOTE: Refer to the chip-specific ERM information for details on Memory 17 mapping. 0 Interrupt notification of Memory 17 non-correctable error events is disabled. 1 Interrupt notification of Memory 17 non-correctable error events is enabled.
6–7 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
8 ESCIE18	Enable Memory 18 Single Correction Interrupt Notification This field is initialized by hardware reset. NOTE: Refer to the chip-specific ERM information for details on Memory 18 mapping. 0 Interrupt notification of Memory 18 single-bit correction events is disabled. 1 Interrupt notification of Memory 18 single-bit correction events is enabled.
9 ENCIE18	Enable Memory 18 Non-Correctable Interrupt Notification This field is initialized by hardware reset. NOTE: Refer to the chip-specific ERM information for details on Memory 18 mapping. 0 Interrupt notification of Memory 18 non-correctable error events is disabled. 1 Interrupt notification of Memory 18 non-correctable error events is enabled.
10–11 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
12 ESCIE19	Enable Memory 19 Single Correction Interrupt Notification This field is initialized by hardware reset. NOTE: Refer to the chip-specific ERM information for details on Memory 19 mapping. 0 Interrupt notification of Memory 19 single-bit correction events is disabled. 1 Interrupt notification of Memory 19 single-bit correction events is enabled.

Table continues on the next page...

ERM_CR2 field descriptions (continued)

Field	Description
13 ENCIE19	Enable Memory 19 Non-Correctable Interrupt Notification This field is initialized by hardware reset. NOTE: Refer to the chip-specific ERM information for details on Memory 19 mapping. 0 Interrupt notification of Memory 19 non-correctable error events is disabled. 1 Interrupt notification of Memory 19 non-correctable error events is enabled.
14–15 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
16–17 Reserved	This field is reserved.
18–19 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
20–21 Reserved	This field is reserved.
22–23 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
24–25 Reserved	This field is reserved.
26–27 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
28–29 Reserved	This field is reserved.
30–31 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

18.3.4 ERM Status Register (ERM_SR0)

The ERM Status Register is a 32-bit control register for signaling which types of ECC events have been detected for each memory channel. The register signals the last memory event to be detected for memory channels 0 to 7.

Address: 0h base + 10h offset = 10h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	SBC0	NCE0	0		SBC1	NCE1	0		SBC2	NCE2	0		SBC3	NCE3	0	
W	w1c	w1c			w1c	w1c			w1c	w1c			w1c	w1c		
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	SBC4	NCE4	0		SBC5	NCE5	0		SBC6	NCE6	0		SBC7	NCE7	0	
W	w1c	w1c			w1c	w1c			w1c	w1c			w1c	w1c		
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

ERM_SR0 field descriptions

Field	Description
0 SBC0	Memory 0 Single-Bit Correction Event This field is initialized by hardware reset. Write 1 to clear this field. This write also clears the corresponding interrupt notification if ERM_CR0[ESCIE0] is enabled. NOTE: Refer to the chip-specific ERM information for details on Memory 0 mapping. 0 No single-bit correction event on Memory 0 detected 1 Single-bit correction event on Memory 0 detected
1 NCE0	Memory 0 Non-Correctable Error Event This field is initialized by hardware reset. Write 1 to clear this field. This write also clears the corresponding interrupt notification if ERM_CR0[ENCIE0] is enabled. NOTE: Refer to the chip-specific ERM information for details on Memory 0 mapping.

Table continues on the next page...

ERM_SR0 field descriptions (continued)

Field	Description
	0 No non-correctable error event on Memory 0 detected 1 Non-correctable error event on Memory 0 detected
2–3 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
4 SBC1	Memory 1 Single-Bit Correction Event This field is initialized by hardware reset. Write 1 to clear this field. This write also clears the corresponding interrupt notification if ERM_CR0[ESCIE1] is enabled. NOTE: Refer to the chip-specific ERM information for details on Memory 1 mapping. 0 No single-bit correction event on Memory 1 detected 1 Single-bit correction event on Memory 1 detected
5 NCE1	Memory 1 Non-Correctable Error Event This field is initialized by hardware reset. Write 1 to clear this field. This write also clears the corresponding interrupt notification if ERM_CR0[ENCIE1] is enabled. NOTE: Refer to the chip-specific ERM information for details on Memory 1 mapping. 0 No non-correctable error event on Memory 1 detected 1 Non-correctable error event on Memory 1 detected
6–7 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
8 SBC2	Memory 2 Single-Bit Correction Event This field is initialized by hardware reset. Write 1 to clear this field. This write also clears the corresponding interrupt notification if ERM_CR0[ESCIE2] is enabled. NOTE: Refer to the chip-specific ERM information for details on Memory 2 mapping. 0 No single-bit correction event on Memory 2 detected 1 Single-bit correction event on Memory 2 detected
9 NCE2	Memory 2 Non-Correctable Error Event This field is initialized by hardware reset. Write 1 to clear this field. This write also clears the corresponding interrupt notification if ERM_CR0[ENCIE2] is enabled. NOTE: Refer to the chip-specific ERM information for details on Memory 2 mapping. 0 No non-correctable error event on Memory 2 detected 1 Non-correctable error event on Memory 2 detected
10–11 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
12 SBC3	Memory 3 Single-Bit Correction Event This field is initialized by hardware reset. Write 1 to clear this field. This write also clears the corresponding interrupt notification if ERM_CR0[ESCIE3] is enabled.

Table continues on the next page...

ERM_SR0 field descriptions (continued)

Field	Description
	NOTE: Refer to the chip-specific ERM information for details on Memory 3 mapping. 0 No single-bit correction event on Memory 3 detected 1 Single-bit correction event on Memory 3 detected
13 NCE3	Memory 3 Non-Correctable Error Event This field is initialized by hardware reset. Write 1 to clear this field. This write also clears the corresponding interrupt notification if ERM_CR0[ENCIE3] is enabled. NOTE: Refer to the chip-specific ERM information for details on Memory 3 mapping. 0 No non-correctable error event on Memory 3 detected 1 Non-correctable error event on Memory 3 detected
14–15 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
16 SBC4	Memory 4 Single-Bit Correction Event This field is initialized by hardware reset. Write 1 to clear this field. This write also clears the corresponding interrupt notification if ERM_CR0[ESCIE4] is enabled. NOTE: Refer to the chip-specific ERM information for details on Memory 4 mapping. 0 No single-bit correction event on Memory 4 detected 1 Single-bit correction event on Memory 4 detected
17 NCE4	Memory 4 Non-Correctable Error Event This field is initialized by hardware reset. Write 1 to clear this field. This write also clears the corresponding interrupt notification if ERM_CR0[ENCIE4] is enabled. NOTE: Refer to the chip-specific ERM information for details on Memory 4 mapping. 0 No non-correctable error event on Memory 4 detected 1 Non-correctable error event on Memory 4 detected
18–19 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
20 SBC5	Memory 5 Single-Bit Correction Event This field is initialized by hardware reset. Write 1 to clear this field. This write also clears the corresponding interrupt notification if ERM_CR0[ESCIE5] is enabled. NOTE: Refer to the chip-specific ERM information for details on Memory 5 mapping. 0 No single-bit correction event on Memory 5 detected 1 Single-bit correction event on Memory 5 detected
21 NCE5	Memory 5 Non-Correctable Error Event This field is initialized by hardware reset. Write 1 to clear this field. This write also clears the corresponding interrupt notification if ERM_CR0[ENCIE5] is enabled.

Table continues on the next page...

ERM_SR0 field descriptions (continued)

Field	Description
	NOTE: Refer to the chip-specific ERM information for details on Memory 5 mapping. 0 No non-correctable error event on Memory 5 detected 1 Non-correctable error event on Memory 5 detected
22–23 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
24 SBC6	Memory 6 Single-Bit Correction Event This field is initialized by hardware reset. Write 1 to clear this field. This write also clears the corresponding interrupt notification if ERM_CR0[ESCIE6] is enabled. NOTE: Refer to the chip-specific ERM information for details on Memory 6 mapping. 0 No single-bit correction event on Memory 6 detected 1 Single-bit correction event on Memory 6 detected
25 NCE6	Memory 6 Non-Correctable Error Event This field is initialized by hardware reset. Write 1 to clear this field. This write also clears the corresponding interrupt notification if ERM_CR0[ENCIE6] is enabled. NOTE: Refer to the chip-specific ERM information for details on Memory 6 mapping. 0 No non-correctable error event on Memory 6 detected 1 Non-correctable error event on Memory 6 detected
26–27 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
28 SBC7	Memory 7 Single-Bit Correction Event This field is initialized by hardware reset. Write 1 to clear this field. This write also clears the corresponding interrupt notification if ERM_CR0[ESCIE7] is enabled. NOTE: Refer to the chip-specific ERM information for details on Memory 7 mapping. 0 No single-bit correction event on Memory 7 detected 1 Single-bit correction event on Memory 7 detected
29 NCE7	Memory 7 Non-Correctable Error Event This field is initialized by hardware reset. Write 1 to clear this field. This write also clears the corresponding interrupt notification if ERM_CR0[ENCIE7] is enabled. NOTE: Refer to the chip-specific ERM information for details on Memory 7 mapping. 0 No non-correctable error event on Memory 7 detected 1 Non-correctable error event on Memory 7 detected
30–31 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

18.3.5 ERM Status Register (ERM_SR1)

The ERM Status Register is a 32-bit control register for signaling which types of ECC events have been detected for each memory channel. The register signals the last memory event to be detected for memory channels 8 to 15.

Address: 0h base + 14h offset = 14h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	SBC8	NCE8	0		SBC9	NCE9	0		SBC10	NCE10	0		SBC11	NCE11	0	
W	w1c	w1c			w1c	w1c			w1c	w1c			w1c	w1c		
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	SBC12	NCE12	0		SBC13	NCE13	0		SBC14	NCE14	0		SBC15	NCE15	0	
W	w1c	w1c			w1c	w1c			w1c	w1c			w1c	w1c		
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

ERM_SR1 field descriptions

Field	Description
0 SBC8	Memory 8 Single-Bit Correction Event This field is initialized by hardware reset. Write 1 to clear this field. This write also clears the corresponding interrupt notification if ERM_CR1[ESCIE8] is enabled. NOTE: Refer to the chip-specific ERM information for details on Memory 8 mapping. 0 No single-bit correction event on Memory 8 detected 1 Single-bit correction event on Memory 8 detected
1 NCE8	Memory 8 Non-Correctable Error Event This field is initialized by hardware reset. Write 1 to clear this field. This write also clears the corresponding interrupt notification if ERM_CR1[ENCIE8] is enabled. NOTE: Refer to the chip-specific ERM information for details on Memory 8 mapping.

Table continues on the next page...

ERM_SR1 field descriptions (continued)

Field	Description
	0 No non-correctable error event on Memory 8 detected 1 Non-correctable error event on Memory 8 detected
2–3 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
4 SBC9	Memory 9 Single-Bit Correction Event This field is initialized by hardware reset. Write 1 to clear this field. This write also clears the corresponding interrupt notification if ERM_CR1[ESCIE9] is enabled. NOTE: Refer to the chip-specific ERM information for details on Memory 9 mapping. 0 No single-bit correction event on Memory 9 detected 1 Single-bit correction event on Memory 9 detected
5 NCE9	Memory 9 Non-Correctable Error Event This field is initialized by hardware reset. Write 1 to clear this field. This write also clears the corresponding interrupt notification if ERM_CR1[ENCIE9] is enabled. NOTE: Refer to the chip-specific ERM information for details on Memory 9 mapping. 0 No non-correctable error event on Memory 9 detected 1 Non-correctable error event on Memory 9 detected
6–7 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
8 SBC10	Memory 10 Single-Bit Correction Event This field is initialized by hardware reset. Write 1 to clear this field. This write also clears the corresponding interrupt notification if ERM_CR1[ESCIE10] is enabled. NOTE: Refer to the chip-specific ERM information for details on Memory 10 mapping. 0 No single-bit correction event on Memory 10 detected 1 Single-bit correction event on Memory 10 detected
9 NCE10	Memory 10 Non-Correctable Error Event This field is initialized by hardware reset. Write 1 to clear this field. This write also clears the corresponding interrupt notification if ERM_CR1[ENCIE10] is enabled. NOTE: Refer to the chip-specific ERM information for details on Memory 10 mapping. 0 No non-correctable error event on Memory 10 detected 1 Non-correctable error event on Memory 10 detected
10–11 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
12 SBC11	Memory 11 Single-Bit Correction Event This field is initialized by hardware reset. Write 1 to clear this field. This write also clears the corresponding interrupt notification if ERM_CR1[ESCIE11] is enabled.

Table continues on the next page...

ERM_SR1 field descriptions (continued)

Field	Description
	NOTE: Refer to the chip-specific ERM information for details on Memory 11 mapping. 0 No single-bit correction event on Memory 11 detected 1 Single-bit correction event on Memory 11 detected
13 NCE11	Memory 11 Non-Correctable Error Event This field is initialized by hardware reset. Write 1 to clear this field. This write also clears the corresponding interrupt notification if ERM_CR1[ENCIE11] is enabled. NOTE: Refer to the chip-specific ERM information for details on Memory 11 mapping. 0 No non-correctable error event on Memory 11 detected 1 Non-correctable error event on Memory 11 detected
14–15 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
16 SBC12	Memory 12 Single-Bit Correction Event This field is initialized by hardware reset. Write 1 to clear this field. This write also clears the corresponding interrupt notification if ERM_CR1[ESCIE12] is enabled. NOTE: Refer to the chip-specific ERM information for details on Memory 12 mapping. 0 No single-bit correction event on Memory 12 detected 1 Single-bit correction event on Memory 12 detected
17 NCE12	Memory 12 Non-Correctable Error Event This field is initialized by hardware reset. Write 1 to clear this field. This write also clears the corresponding interrupt notification if ERM_CR1[ENCIE12] is enabled. NOTE: Refer to the chip-specific ERM information for details on Memory 12 mapping. 0 No non-correctable error event on Memory 12 detected 1 Non-correctable error event on Memory 12 detected
18–19 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
20 SBC13	Memory 13 Single-Bit Correction Event This field is initialized by hardware reset. Write 1 to clear this field. This write also clears the corresponding interrupt notification if ERM_CR1[ESCIE13] is enabled. NOTE: Refer to the chip-specific ERM information for details on Memory 13 mapping. 0 No single-bit correction event on Memory 13 detected 1 Single-bit correction event on Memory 13 detected
21 NCE13	Memory 13 Non-Correctable Error Event This field is initialized by hardware reset. Write 1 to clear this field. This write also clears the corresponding interrupt notification if ERM_CR1[ENCIE13] is enabled.

Table continues on the next page...

ERM_SR1 field descriptions (continued)

Field	Description
	NOTE: Refer to the chip-specific ERM information for details on Memory 13 mapping. 0 No non-correctable error event on Memory 13 detected 1 Non-correctable error event on Memory 13 detected
22–23 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
24 SBC14	Memory 14 Single-Bit Correction Event This field is initialized by hardware reset. Write 1 to clear this field. This write also clears the corresponding interrupt notification if ERM_CR1[ESCIE14] is enabled. NOTE: Refer to the chip-specific ERM information for details on Memory 14 mapping. 0 No single-bit correction event on Memory 14 detected 1 Single-bit correction event on Memory 14 detected
25 NCE14	Memory 14 Non-Correctable Error Event This field is initialized by hardware reset. Write 1 to clear this field. This write also clears the corresponding interrupt notification if ERM_CR1[ENCIE14] is enabled. NOTE: Refer to the chip-specific ERM information for details on Memory 14 mapping. 0 No non-correctable error event on Memory 14 detected 1 Non-correctable error event on Memory 14 detected
26–27 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
28 SBC15	Memory 15 Single-Bit Correction Event This field is initialized by hardware reset. Write 1 to clear this field. This write also clears the corresponding interrupt notification if ERM_CR1[ESCIE15] is enabled. NOTE: Refer to the chip-specific ERM information for details on Memory 15 mapping. 0 No single-bit correction event on Memory 15 detected 1 Single-bit correction event on Memory 15 detected
29 NCE15	Memory 15 Non-Correctable Error Event This field is initialized by hardware reset. Write 1 to clear this field. This write also clears the corresponding interrupt notification if ERM_CR1[ENCIE15] is enabled. NOTE: Refer to the chip-specific ERM information for details on Memory 15 mapping. 0 No non-correctable error event on Memory 15 detected 1 Non-correctable error event on Memory 15 detected
30–31 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

18.3.6 ERM Status Register (ERM_SR2)

The ERM Status Register is a 32-bit control register for signaling which types of ECC events have been detected for each memory channel. The register signals the last memory event to be detected for memory channels 16 to 19.

Address: 0h base + 18h offset = 18h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	SBC16	NCE16	0		SBC17	NCE17	0		SBC18	NCE18	0		SBC19	NCE19	0	
W	w1c	w1c			w1c	w1c			w1c	w1c			w1c	w1c		
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

ERM_SR2 field descriptions

Field	Description
0 SBC16	Memory 16 Single-Bit Correction Event This field is initialized by hardware reset. Write 1 to clear this field. This write also clears the corresponding interrupt notification if ERM_CR2[ESCIE16] is enabled. NOTE: Refer to the chip-specific ERM information for details on Memory 16 mapping. 0 No single-bit correction event on Memory 16 detected 1 Single-bit correction event on Memory 16 detected
1 NCE16	Memory 16 Non-Correctable Error Event This field is initialized by hardware reset. Write 1 to clear this field. This write also clears the corresponding interrupt notification if ERM_CR2[ENCIE16] is enabled. NOTE: Refer to the chip-specific ERM information for details on Memory 16 mapping.

Table continues on the next page...

ERM_SR2 field descriptions (continued)

Field	Description
	0 No non-correctable error event on Memory 16 detected 1 Non-correctable error event on Memory 16 detected
2–3 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
4 SBC17	Memory 17 Single-Bit Correction Event This field is initialized by hardware reset. Write 1 to clear this field. This write also clears the corresponding interrupt notification if ERM_CR2[ESCIE17] is enabled. NOTE: Refer to the chip-specific ERM information for details on Memory 17 mapping. 0 No single-bit correction event on Memory 17 detected 1 Single-bit correction event on Memory 17 detected
5 NCE17	Memory 17 Non-Correctable Error Event This field is initialized by hardware reset. Write 1 to clear this field. This write also clears the corresponding interrupt notification if ERM_CR2[ENCIE17] is enabled. NOTE: Refer to the chip-specific ERM information for details on Memory 17 mapping. 0 No non-correctable error event on Memory 17 detected 1 Non-correctable error event on Memory 17 detected
6–7 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
8 SBC18	Memory 18 Single-Bit Correction Event This field is initialized by hardware reset. Write 1 to clear this field. This write also clears the corresponding interrupt notification if ERM_CR2[ESCIE18] is enabled. NOTE: Refer to the chip-specific ERM information for details on Memory 18 mapping. 0 No single-bit correction event on Memory 18 detected 1 Single-bit correction event on Memory 18 detected
9 NCE18	Memory 18 Non-Correctable Error Event This field is initialized by hardware reset. Write 1 to clear this field. This write also clears the corresponding interrupt notification if ERM_CR2[ENCIE18] is enabled. NOTE: Refer to the chip-specific ERM information for details on Memory 18 mapping. 0 No non-correctable error event on Memory 18 detected 1 Non-correctable error event on Memory 18 detected
10–11 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
12 SBC19	Memory 19 Single-Bit Correction Event This field is initialized by hardware reset. Write 1 to clear this field. This write also clears the corresponding interrupt notification if ERM_CR2[ESCIE19] is enabled.

Table continues on the next page...

ERM_SR2 field descriptions (continued)

Field	Description
	NOTE: Refer to the chip-specific ERM information for details on Memory 19 mapping. 0 No single-bit correction event on Memory 19 detected 1 Single-bit correction event on Memory 19 detected
13 NCE19	Memory 19 Non-Correctable Error Event This field is initialized by hardware reset. Write 1 to clear this field. This write also clears the corresponding interrupt notification if ERM_CR2[ENCIE19] is enabled. NOTE: Refer to the chip-specific ERM information for details on Memory 19 mapping. 0 No non-correctable error event on Memory 19 detected 1 Non-correctable error event on Memory 19 detected
14–31 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

18.3.7 ERM Memory n Error Address Register (ERM_EAR n)

The ERM_EAR n is a 32-bit register for capturing the address of the last ECC event in Memory n , where n denotes the memory channel. Any attempted write to ERM_EAR n is ignored.

Address: 0h base + 100h offset + (16d × i), where i=0d to 19d

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	EAR																															
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	

ERM_EAR n field descriptions

Field	Description
0–31 EAR	Memory n Error Address - This field contains the faulting system address of the last recorded ECC event on Memory n . NOTE: Refer to the chip-specific ERM information for details on Memory n mapping.

18.3.8 ERM Memory n Syndrome Register (ERM_SYN n)

The ERM_SYN n is a 32-bit register for capturing the calculated syndrome of the last ECC event on Memory n , where n denotes the memory channel. Any attempted write to ERM_SYN n is ignored. The syndrome value identifies the pertinent bit position on a correctable, single-bit data inversion or a non-correctable, single-bit address inversion. The syndrome value does not provide any additional diagnostic information on non-correctable, multi-bit inversions.

Address: 0h base + 104h offset + (16d × i), where i=0d to 19d

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	SYNDROME								0																							
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

ERM_SYN n field descriptions

Field	Description
0–7 SYNDROME	Memory n Syndrome - This field contains the ECC syndrome associated with the last recorded ECC event on Memory n . NOTE: Refer to the chip-specific ERM information for details on Memory n mapping.
8–31 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

18.4 Functional description

18.4.1 Single-bit correction events

When a single-bit correction event on Memory n is detected, the ERM:

- Records the event by changing to 1 the value of the applicable Status Register bit: ERM_SR x [SBC n]
- Records the corresponding access address that initiated the event in the Memory n Error Address Register: ERM_EAR n
- Stores the corresponding ECC syndrome in the Memory n Error Syndrome Register: ERM_SYN n . This register identifies the bit position of the corrected data on single-bit data inversion.

The ERM holds event information only for the last reported event.

To clear the record of an event, write 1 to ERM_SR x [SBC n] to change its value to 0.

18.4.1.1 Optional interrupt notification for single-bit correction events

The ERM provides an option to generate an interrupt notification upon the report of a single-bit correction event. This sequence describes how to use the option:

1. To enable interrupt notifications for single-bit correction events on Memory n , set ERM_CR x [ESCIEn] to 1.
2. Subsequently, when a single-bit correction event on Memory n is detected, the ERM:
 - Records the event and address and stores the ECC syndrome as usual
 - Additionally sends an interrupt notification corresponding to the event
3. To clear both the record of an event and the corresponding interrupt notification, write 1 to ERM_SR x [SBC n] to change its value to 0.

18.4.2 Non-correctable error events

When a non-correctable ECC error event on Memory n is detected, the ERM:

- Records the event by changing to 1 the value of the applicable Status Register bit: ERM_SR x [NCE n]
- Records the corresponding access address that initiated the event in the Memory n Error Address Register: ERM_EAR n
- Stores the corresponding ECC syndrome in the Memory n Error Syndrome Register: ERM_SYN n
 - In the event of a non-correctable address bit inversion, ERM_SYN n identifies the pertinent address bit position.
 - In the event of a non-correctable, multi-bit data inversion, the syndrome value does not provide any additional diagnostic information.

The ERM holds event information only for the last reported event.

To clear the record of an event, write 1 to ERM_SR x [NCE n] to change its value to 0.

18.4.2.1 Optional interrupt notification for non-correctable error events

The ERM provides an option to generate an interrupt notification upon the report of a non-correctable ECC event. This sequence describes how to use the option:

1. To enable interrupt notifications for non-correctable error events on Memory n , set `ERM_CRx[ENCIE $n$ ]` to 1.
2. Subsequently, when a non-correctable error event on Memory n is detected, the ERM:
 - Records the event and address and stores the ECC syndrome as usual
 - Additionally sends an interrupt notification corresponding to the event
3. To clear both the record of an event and the corresponding interrupt notification, write 1 to `ERM_SRx[NCE $n$ ]` to change its value to 0.

18.5 Initialization

For each ERM channel, prepare the corresponding memory array before enabling ERM interrupts about errors for that memory.

1. Initialize the memory to a known value so that the correct corresponding ECC codeword is stored.
2. During the memory's initialization, if the ERM captures information about any ECC error event, clear the corresponding `ERM_SRx[SBC $n$ ]` or `ERM_SRx[NCE $n$ ]` field that stores the record of the event.
3. Program the applicable `ERM_CRx[ESCI $n$ ]` and `ERM_CRx[ENCIE $n$ ]` fields to enable ERM interrupts as desired.

Chapter 19

Clocking

19.1 Introduction

This chapter describes the architecture for the system level clocks and includes the following information:

- System clock specifics
- Clock architecture
- Clock sources
- Clock monitoring
- Programmable clock dividers
- Clock control registers

The clock architecture of this device contains two clock domains coming from separate PLLs. The first one is the system clock for the cores, cross-bar switch, peripheral bridges, memories, debug logic and memory mapped portion of peripherals. The second one feeds the state machines and protocol engines of communication and timer peripheral modules. These two clock domains can be configured to be completely asynchronous between each other. The PLL generating the system clock (PLL1) can be configured to be frequency modulated to reduce electromagnetic emissions. The PLL generating the peripheral clock (PLL0) cannot be modulated. It is intended to provide an accurate reference clock for peripheral timer and communication functions that cannot tolerate frequency modulation.

The reference clock for PLL0 (non-FM) may come from the following possible sources:

- Crystal oscillator (XOSC)
- External clock (when XOSC is bypassed)
- Internal 16 MHz RC oscillator (IRC)

The reference clock for PLL1 (FM-modulated) may come from the following possible sources:

- Crystal oscillator (XOSC)
- External clock (when XOSC is bypassed)
- Output of PLL0

Out of reset, both clock domains (system and peripheral) are sourced directly from the IRC. After reset, software can select the PLLs or other clock sources (XOSC, external) to connect to system and peripheral clocks. A Progressive Clock Frequency Switch (PCFS) module is available after the system clock selector to allow gradual frequency changes that minimize transient currents. Clock Monitor Units (CMUs) are connected to outputs of clock dividers to verify that their clock frequencies stay within necessary operating limits. When an issue with a clock is found, the CMU sends a failure signal directly to the FCCU.

19.2 Clock generation

The top-level clock generation architecture can be seen in the figure below. All clock selectors and dividers are located and programmed in memory mapped registers of the SIU.

Notes: 1. The clock divider range in the PLL outputs (phi) is non-contiguous in by-2 steps: [2, 4, ..., 124, 126].

2. Divider values and mux selectors are located in SIU registers:
SIU_SYSDIV, SIU_SDCLKCFG, SIU_LFCLKCFG, SIU_PSCLKCFG

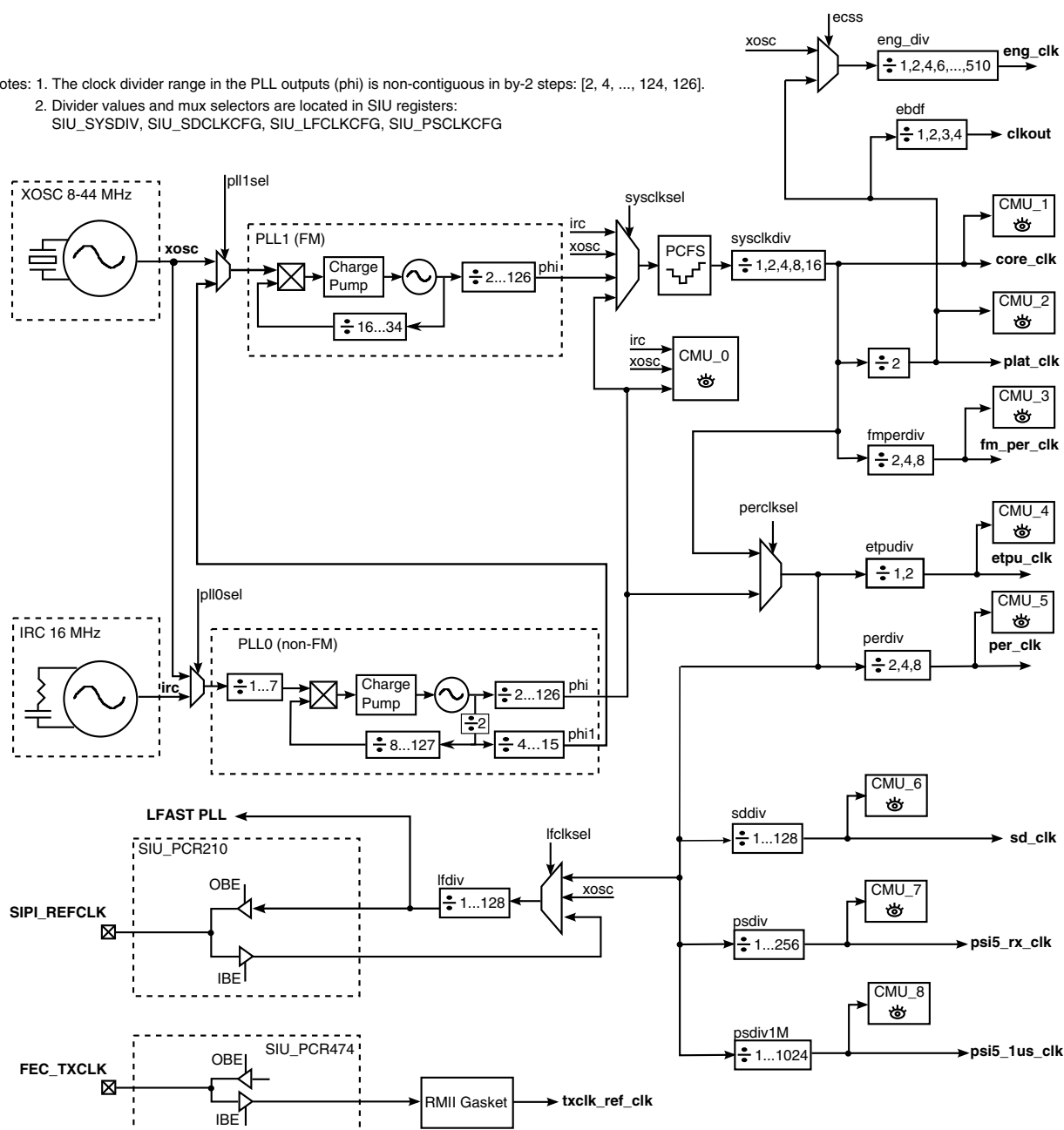


Figure 19-1. Clock generation diagram

19.3 SIU registers

The following table shows the relationship between the clocks of the clock generation diagram and the SIU registers.

Table 19-1. Relationship between clocks and SIU registers

Clock	Clock destination	Clock selector	Clock divider	CMU
PLL0	PLL0 input	SIU_SYSDIV.PLL0SEL	Internal to PLL0	
PLL1	PLL1 input	SIU_SYSDIV.PLL1SEL	Internal to PLL1	
core_clk	Core0, Core1, Checker Core	SIU_SYSDIV.SYSCLKSEL	SIU_SYSDIV.SYSCLKDIV	CMU_1 ¹
plat_clk	Platform modules: Cross-Bar, Peripheral Bridges, Memories, DMA, Flash Controller, Debug		Fixed at divided-by-2	CMU_2 ¹
fm_per_clk	Memory mapped registers of peripherals		SIU_SYSDIV.FMPERDIV	CMU_3
per_clk	Protocol and communication engines of peripherals	SIU_SYSDIV.PERCLKSEL	SIU_SYSDIV.PERDIV	CMU_5
etpu_clk	eTPU A/B/C		SIU_SYSDIV.ETPUDIV	CMU_4
sd_clk	Sigma-Delta ADC		SIU_SDCLKCFG.SDDIV	CMU_6
psi5_rx_clk	PSI5 Rx clock input		SIU_PSCLKCFG.PSDIV	CMU_7
psi5_1us_clk	PSI5 1 MHz clock input		SIU_PSCLKCFG.PSDIV1M	CMU_8
LFAST PLL	LFAST PLL input	SIU_LFCLKCFG.LFCLKSEL	SIU_LFCLKCFG.LFDIV	
MCAN	MCAN Protocol Engine Clock	SIU_SYSDIV.MCANSEL	Internal to MCAN	
clkout	EBI CLKOUT pin		SIU_ECCR.EBDF	
eng_clk	Engineering clock pin	SIU_ECCR.ECSS	SIU_ECCR.ENGDIV	

1. CMU_1 and CMU_2 are physically connected to clock tree spots inside the checker core lake.

19.4 Operating frequencies

The device supports the maximum operating frequencies given in the following table.

Table 19-2. Maximum system and peripheral clock frequencies

Clock domain	Mode	Core (CORE_CLK)	eTPU (eTPU_CLK)	Platform (PLAT_CLK)	Peripherals memory mapped registers (FM_PER_CLK)	Peripherals protocol engines (PER_CLK)
Backwards compatible: one clock domain ¹	Double	264	132	132	132	132
	eTPU	200	200	100	100	100
Enhanced: dual clock domain ²	Normal	264	200	132	132	100

1. Clock selectors configured such that all clocks come from a single source, either PLL0 or PLL1.

2. Clock selectors configured such that both PLLs drive the clocks into two asynchronous domains.

NOTE

The essentially asynchronous clock domains of EVTO (eTPU_CLK) and NPC (CORE_CLK) require CORE_CLK to be set to a higher frequency than eTPU_CLK for guaranteed

EVTO capture by NPC. Because CORE_CLK can be set to FM (2% intentional jitter), the frequency of CORE_CLK minus 2% must be greater than the frequency of eTPU_CLK.

19.5 Default clock configuration

At power up, the IRC is the default clock for the entire system, with clock selectors configured such that all clocks come from a single clock domain. The default configuration after reset for all clock sources, selectors, and dividers is shown in the tables below.

Table 19-3. Reset configuration of clock sources

Source	Default after reset
IRC	Always enabled; it is the default clock source.
XOSC/External clock	Crystal oscillator and external bypass default state depends on DCF record or configuration pins (see SIU_RSR subsection of the SIU chapter).
PLL0	Disabled and powered down
PLL1	Disabled and powered down

Table 19-4. Reset configuration of clock selectors

Selector	Default after reset
SIU_SYSDIV.PLL0SEL	Selects XOSC
SIU_SYSDIV.PLL1SEL	Selects XOSC
SIU_SYSDIV.SYSCLKSEL	Selects the IRC
SIU_SYSDIV.PERCLKSEL	Selects core_clk
SIU_LFCLKCFG.LFCLKSEL	Selects the XOSC
SIU_SYSDIV.MCANSEL	Selects the XOSC
SIU_ECCR.ECSS	Selects plat_clk

Table 19-5. Reset configuration of clock dividers

Divider	Default after reset
SIU_SYSDIV.SYSCLKDIV	Divide-by-1
SIUSYSDIV.FMPERDIV	Divide-by-2
SIU_SYSDIV.PERDIV	Divide-by-2
SIU_SYSDIV.ETPUDIV	Divide-by-2
SIU_SDCLKCFG.SDDIV	Divide-by-1
SIU_PSCLKCFG.PSDIV	Divide-by-1

Table continues on the next page...

Table 19-5. Reset configuration of clock dividers (continued)

Divider	Default after reset
SIU_PSCLKCFG.PSDIV1M	Divide-by-1
SIU_LFCLKCFG.LFDIV	Divide-by-1
SIU_ECCR.EBDF	Divide-by-2
SIU_ECCR.ENGDIV	Divide-by-32

19.6 Clock sources

This chip has these clock sources:

- 16 MHz Internal RC oscillator (IRC)
- External oscillator (XOSC)
- PLL0 (non-FM PLL)
- PLL1 (FMPLL)

19.6.1 16 MHz RC oscillator (IRC)

This device has a 16 MHz internal RC oscillator that is always enabled and cannot be powered down. The IRC can be used as the source of system clock, peripheral clock and PLL clock references. The IRC is the default system and peripheral clock after reset.

The IRC has no dedicated software accessible register interface.

19.6.2 External oscillator (XOSC)

The external oscillator (XOSC) provides a clock source for the on-chip PLLs and some peripherals that require accurate clock reference (FlexCAN, MCAN, PIT, LFAST PLL, CMU_0). If required, it can also be used directly as a clock source to the main system and peripheral clocks. The XOSC does not have any software accessible register interface. Configuration is done using DCF records read from UTEST flash memory, which are read by the SSCM during reset and driven to the oscillator. Changing the XOSC configuration requires a reset cycle to take effect.

NOTE

The DCF configuration bits for the XOSC have triple-voting flip-flop (TVF) implementation to prevent undetectable errors in the system. See the Device Configuration Format chapter for more information on DCF programming.

19.6.2.1 XOSC configuration

The XOSC has the following configurable features:

- Crystal oscillator can be bypassed to source a clock directly from the EXTAL pin (bypass mode)
- Internal load capacitors can be selected for 20/40 MHz crystal oscillators
- Automatic Loop Control (ALC) can be applied to avoid the need of external series resistor
- Oscillator gain can be selected to match the crystal frequency range

Bypass configuration is controlled by bit 24 of the Miscellaneous DCF record. The selection of internal or external load capacitors on the XTAL/EXTAL pins is determined by bit 25 of the Miscellaneous DCF record. Load capacitor values are determined by the crystal manufacturer data sheet requirements, while accounting for stray PCB and on-chip capacitance. See the device data sheet for on-chip capacitance values.

When using internal load capacitors for the oscillator, the internal capacitance value is configured in the range of 1–16 pF, in approximate steps of 1 pF, by bits 26–30 of the Miscellaneous DCF record. The default value for the trimming bits (0b00000) selects the minimum internal capacitance value (about 1.8 pF).

The XOSC has the ability to be started with a crystal of 8 MHz, 20 MHz, or 40 MHz. Bits 18 and 21 of the Miscellaneous DCF record select which crystal source to use.

Automatic Loop Control (ALC) can be enabled by bit 17 of the Miscellaneous DCF record.

19.6.2.2 XOSC startup

During reset, the SSCM reads the XOSC_CNT field from the Miscellaneous DCF record and pre-loads this value to an internal counter clocked by XOSC. Upon reset exit, the counter counts 512 times XOSC_CNT. When counting finishes, it means that the XOSC clock is considered to be stable, and a flag called XOSC is set in the SIU_RSR register. This is a “set-once” bit, which software can check before attempting to use the external oscillator clock source soon after a reset cycle. This flag also enables CMU0 to start

monitoring the XOSC clock. From this moment on, if the XTAL clock fails, the SIU_RSR[XOSC] flag will not be negated to indicate the fail condition, but CMU0 will set a flag and also send an event signal to the FCCU.

19.6.3 PLL

This chip has a dual PLL subsystem that provides separate system and peripheral clocks.

The PLLs are disabled after reset. Enable them by software before selecting them as clock sources.

The following image is a block diagram of the dual PLL subsystem.

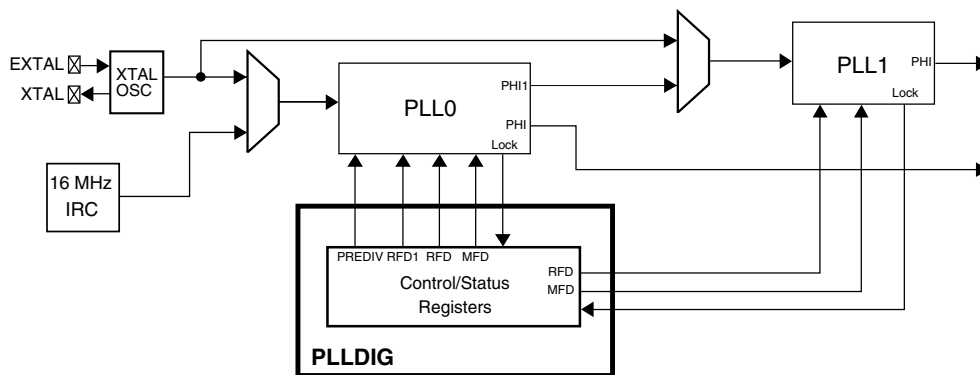


Figure 19-2. Dual PLL digital interface block diagram

19.6.3.1 PLL0 (non-FM)

PLL0 is the primary PLL. It provides a non-Frequency Modulated clock to the peripherals and the reference clock to PLL1.

19.6.3.1.1 PLL0 input clocks

The possible input clock sources for PLL0 are the XOSC and IRC. Select the clock reference using multiplexers controlled by the SIU_SYSDIV register.

19.6.3.1.2 PLL0 output clocks

The output clocks from PLL0 are PHI and PHI1.

- The PHI output clock drives various peripheral clocks and the system clock when PLL1 is not locked or active.
- The PHI1 output provides one of the input references for PLL1.

To generate the reference clock to PLL1, PLL0 has a dedicated output clock divider that is separate from the divider that generates the clock to peripherals.

19.6.3.2 PLL1 (FM)

PLL1 is a Frequency Modulated PLL (FMPLL) that is used to drive the system clock.

19.6.3.2.1 PLL1 input clocks

The possible input clock sources for PLL1 are PLL0 and the XOSC. Select the clock reference using multiplexers controlled by the SIU_SYSDIV register.

19.6.3.2.2 PLL1 output clocks

The output clock from PLL1 drives the system clock. The PHI output clock contains a fractional divider that can be applied to the loop divide of the PLL to achieve good granularity in the PLL1 PHI output clock frequency.

19.6.3.3 PLL register interface

This chip has a single digital interface for both PLLs. The digital interface provides register control of all available features of PLL0 and PLL1.

Before changing any of the dividers or FM configuration, you must force the PLLs into a disabled state by writing 00b to the CLKCFG field of the PLL0CR and PLL1CR registers. This method is the only way to power down the PLLs.

19.6.3.4 PLLDIG initialization information

Immediately after reset, PLL0 and PLL1 are disabled. The recommended procedure to program the PLLs and enter normal mode is:

1. Configure PLL0 and related modules.
 - a. With PLL0 disabled, program the desired PLL0 clock source in SIU_SYSDIV[PLL0SEL]. The default source is XOSC. Write 1 to SIU_SYSDIV[PLL0SEL] to select IRC as the source.
 - b. Program PLL0DV[PREDIV], PLL0DV[RFDPHI1], PLL0DV[MFD], and PLL0DV[RFDPHI].

- c. If necessary, modify the CMU frequency meter values (CMU_MDR and CMU_FDR) used to monitor the XOSC frequency. The XOSC frequency is compared to the IRCOSC source.
2. Ensure SIU_RSR[XOSC] is 1 to confirm that XOSC is stable.
3. Turn on PLL0.
 - a. Enable PLL0 by setting PLL0CR[CLKCFG] to 11b.
 - b. Poll PLL0SR[LOCK] until its value is 1.
4. Configure PLL1 and related modules.
 - a. With PLL1 disabled, program the PLL1 clock source in SIU_SYSDIV[PLL1SEL]. The default source is XOSC. Write 1 to SIU_SYSDIV[PLL1SEL] to select the PHI1 output of PLL0 as the source.
 - b. Program PLL1DV[MFD] and PLL1DV[RFDPHI].
 - c. If required:
 - Program the PLL1FM register with the desired frequency modulation parameters, and
 - Enable FM modulation by setting PLL1FM[MODEN] to 1.

The PLL1FM register must not be written after PLL1 is enabled and operating in normal mode.
5. Turn on PLL1.
 - a. Enable PLL1 by setting PLL1CR[CLKCFG] to 11b.
 - b. Poll PLL1SR[LOCK] until its value is 1.
6. Source the system clock and peripheral clock from the PLLs by programming SIU_SYSDIV[SYSCLKSEL] and SIU_SYSDIV[PERCLKSEL], respectively.

19.7 Clock gating

The SIU_HLT1 and SIU_HLT2 registers provide software-controlled clock gating. These registers dispatch request signals to various modules to halt their clock inputs. Modules respond differently to the halt request. The CPU allows its clock inputs to be halted only after the execution of the WAIT instruction. Most other modules wait for internal activities to finish before allowing their clock inputs to be halted by asserting a halt

acknowledge signal. A few modules do not implement the stop acknowledge protocol within them and allow their clocks to be halted immediately after the halt request is asserted.

Some peripherals have an internal memory mapped register that can be used to disable the module and to halt the clock input to the module. In most cases the peripheral registers of disabled modules are still readable and writable. However, the halt mechanism of the SIU_HALT registers disables the read/write functions of the peripheral registers for additional power saving.

When the clocks of all CPUs have been halted and all XBAR masters are idle, then the clocks of the following modules are also halted automatically: XBAR, Peripheral Bridges, system RAM, DMA, STM and Semaphores. The INTC and SWT_A/SWT_B clocks are never stopped.

19.8 Core and on-platform clocking

The clocks to the cores and on-platform modules come from the FM PLL. The core clock (CORE_CLK) is double the frequency of the platform clock (PLAT_CLK), but they are synchronous to each other (same clock domain). The frequency ratio between CORE_CLK and PLAT_CLK is fixed at 2:1.

The following modules belong to the core clock domain:

- Core0
- Core1
- Checker core
- Cache memories
- Nexus Port Controller (NPC)

The following modules belong to the platform clock domain:

- System RAM and RAM Controller
- Flash memory and Flash Memory Controller
- Crossbar Switch
- DMA

- SIPI
- EBI

The frequency modulated peripheral clock (FM_PER_CLK) mediates the communication between the Peripheral Bridges and memory-mapped registers within peripheral modules. This clock is part of the same FM clock domain as the core and platform clocks.

19.9 Peripheral clocks

Some modules have two clock domains implemented within them. One domain is for the state machines (protocol and/or timing engines) that require accurate non-FM clock sources, and the other domain is for the memory mapped registers that must be accessed by the Peripheral Bridges at the FM clock domain. These modules with separate clock domains can be connected directly to the two FM and non-FM clock sources.

Some other modules, like the eTPU, have a single clock domain with a single clock input. The following figure depicts all of these cases.

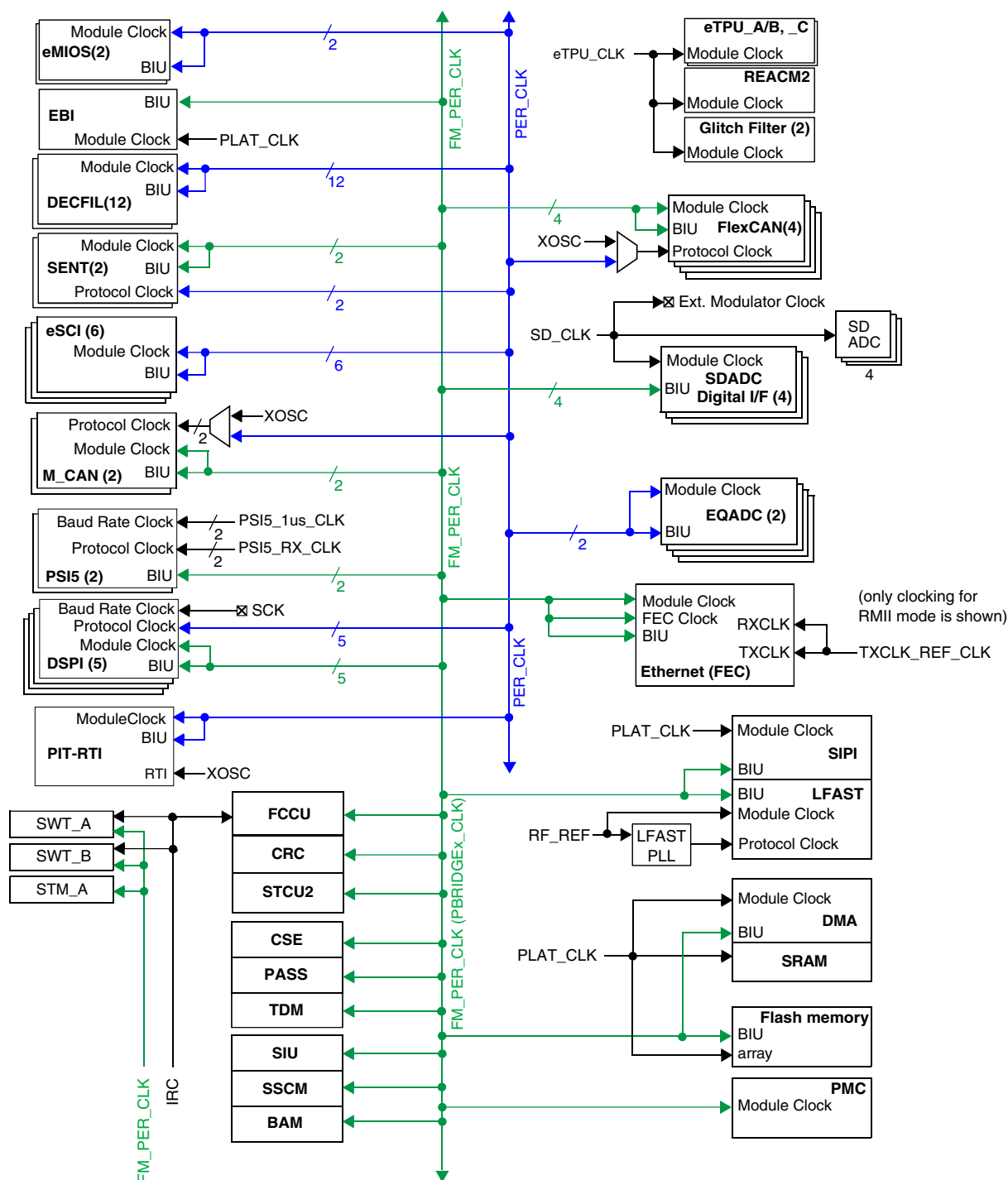


Figure 19-3. Clock distribution

19.9.1 eTPU clocking

The eTPUs have a separate clock domain specific to them (ETPU_CLK), coming from the non-FM PLL. The eTPU clock is synchronous to the other non-FM peripheral clock domain (PER_CLK), but is asynchronous to the FM-domain clocks (CORE_CLK, PLAT_CLK, FM_PER_CLK). The reason for a separate clock domain to the eTPUs is to allow them to run at higher speeds than other peripherals, thus optimizing performance and power.

The eTPU module itself has just one clock input that is used by all internal circuits, including timing engines and memory mapped registers. Because this single clock input is connected to the non-FM clock domain, a synchronization gasket mediates communications between the eTPUs and the peripheral bridges, which run at the FM-modulated clock domain. The operation of this gasket is transparent to software.

19.9.2 FlexCAN and MCAN clocking

The clock source for the CAN protocol engines can optionally come from the peripheral clock or from the XOSC. For the FlexCAN module, selection of the engine clock source is based on a register field internal to the FlexCAN module. Each FlexCAN instance can select a different source.

For the MCAN modules, selection of the engine clock source is based on the SIU_SYSDIV[MCANSEL] field. In this case, all MCAN instances receive the same clock.

19.9.3 LFAST clocking

This chip includes an LFAST module to support interprocessor communications in conjunction with the SIPI module. A PLL dedicated to the LFAST receives a 10 MHz to 20 MHz reference clock and generates the high-speed clock required to support LFAST serial communication. For low speed LFAST operation, the reference clock is used directly by the LFAST module.

The LFAST PLL reference clock source may come from the PLL peripheral clock domain, from the XOSC, or from an external LFAST device via the SIPI_REFCLK package pin. When this reference clock is generated internally, it can be output on the SIPI_REFCLK pin.

NOTE

For the SIPI_REFCLK pin, which SIU_PCR210 configures:

- If SIU_PCR210's IBE and OBE fields are both set to 1, OBE takes precedence and is output on the pin; the external input path of the pin to SIPI does not work.
- The external clock comes in only when IBE is set to 1 and OBE is cleared to 0.

19.9.4 PSI5 clocking

PSI5 requires two clocks, which have their own dividers coming from the peripheral clock domain.

19.9.5 Ethernet clocking

Only an external clock for RMII mode is supported. This chip does not generate an FEC clock internally. The external clock comes from a package pin and goes through the RMII gasket.

NOTE

While the FEC is operating, do not change clock configurations (muxes or dividers). Otherwise, FEC transactions in the crossbar might be corrupted.

19.9.6 SDADC clocking

The SDADC does not support the use of an external modulator. As a result, the only clock available is the internal clock (sd_clk), which is derived from the peripheral (non-FM) clock domain through a clock divider. The division factor for this divider is specified by the SIU_SDCLKCFG[SDDIV] field.

19.10 Serial boot clocking

Because the IRC is not accurate enough for the serial protocols, prior to entering serial boot mode the BAM switches the clock source to the external oscillator, using either the crystal or external reference—whichever one is effective at startup. No direct signal is available to an external host to indicate that serial boot has been initiated by the BAM.

19.11 MCKO

MCKO is an output clock from the Nexus Port Controller (NPC) to the development tools used for the timing of MSEO and MDO pin functions. MCKO is derived from the core clock (core_clk), and its frequency is determined by the value of the MCKO_DIV field in the port configuration register (PCR) located in the NPC.

19.12 CLKOUT / ENGCLK

CLKOUT is a device clock output derived from the platform clock and intended for, but not restricted to, use with the External Bus Interface (EBI). ENGCLK is a 50% duty cycle output clock derived from the platform clock or the crystal oscillator. Both clocks belong to the FM-PLL clock domain, but ENGCLK is not synchronous to CLKOUT (not edge aligned).

The external clock pins CLKOUT and ENGCLK are enabled during and immediately after reset. CLKOUT defaults to the platform clock frequency divided by two, and ENGCLK defaults to the platform clock frequency divided by 32.

19.13 Clock monitoring

For all safety-relevant clocks, this chip uses [Clock Monitor Unit \(CMU\)](#) modules to detect a missing clock or incorrect frequency. The IRC and XOSC are used as the clock monitor references.

The chip has nine instances of the CMU module. Each CMU module is programmed independently.

19.13.1 CMU-monitored clocks

The following table shows the clock or clocks monitored by each CMU module instance.

Table 19-6. CMU-monitored clocks

Monitor	Monitored clock(s)
CMU_0	PLL0, XOSC, IRC
CMU_1	Core0, Core1, Checker Core, RCCU
CMU_2	Platform modules: Crossbar, Peripheral Bridges, Memories, DMA, Flash Memory Controller, Debug

Table continues on the next page...

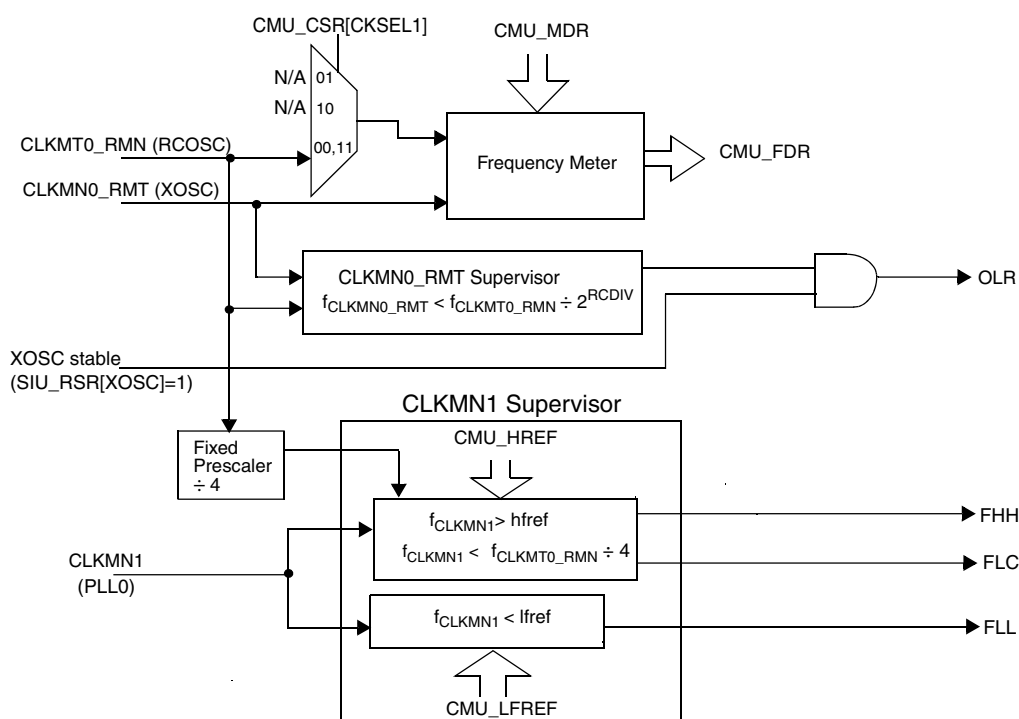
Table 19-6. CMU-monitored clocks (continued)

Monitor	Monitored clock(s)
CMU_3	Memory-mapped registers of peripherals
CMU_4	eTPU_A, eTPU_B, eTPU_C
CMU_5	Protocol and communication engines of peripherals
CMU_6	SDADC
CMU_7	PSI5 Rx clock input
CMU_8	PSI5 1 MHz clock input

19.13.2 CMU_0 configuration

CMU_0 automatically monitors the XOSC clock and is able to monitor PLL0.

The following figure is a block diagram for CMU_0 on this chip.

**Figure 19-4. CMU_0 block diagram**

19.13.2.1 CMU_0 monitoring of XOSC

CMU_0 is the only CMU instance that monitors the XOSC clock. CMU_0 uses the IRC clock to measure whether the XOSC clock's frequency is too slow.

CMU_0's clock monitoring of the XOSC is not controlled by software. Its monitoring of XOSC is always enabled automatically after the SIU_RSR[XOSC] field indicates that the XOSC clock is stable.

19.13.2.2 CMU_0 monitoring of PLL0

To enable CMU_0's monitoring of PLL0, the proper initialization sequence is:

1. Configure and enable PLL0.
2. Wait for PLL0 to lock.
3. Enable CMU_0 for monitoring by setting the CMU_0_CSR[CME] field.

19.13.3 CMU_1 to CMU_8 configuration

For CMU_1 to CMU_8:

- Each CMU operates independently from the module that it monitors, with no hardware interlock between them.
- The IRC is the reference clock.

The following figure is a block diagram for CMU_1 to CMU_5 on this chip.

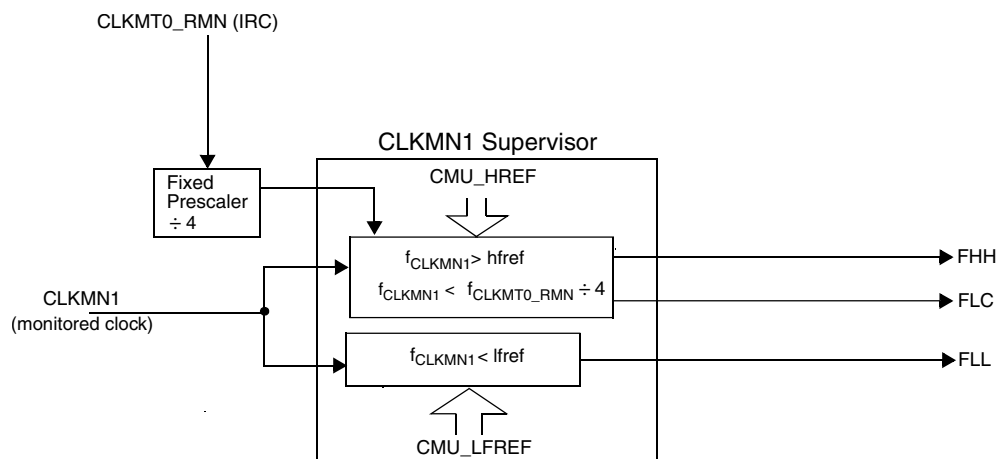


Figure 19-5. CMU_1 to CMU_5 block diagram

The following figure is a block diagram for CMU_6 to CMU_8 on this chip.

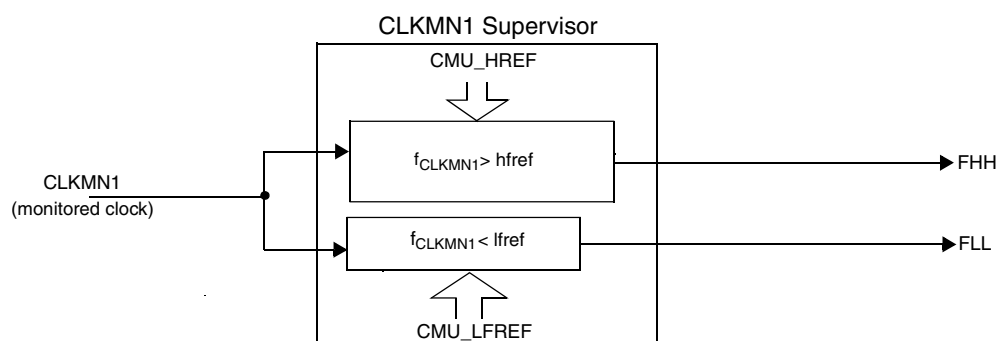


Figure 19-6. CMU_6 to CMU_8 block diagram

To enable monitoring of each of CMU_1 to CMU_8, the proper initialization sequence is:

1. Enable the clock source corresponding to the particular CMU instance.
2. Ensure that the particular clock source is enabled, stable, and ready to be monitored.
 - If the XOSC is the clock source, then the SIU_RSR[XOSC] flag must be set.
 - If one of the PLLs is the clock source, then that PLL must be locked.
3. Enable the particular CMU instance for monitoring by setting the applicable CMU_x_CSR[CME] field.

19.13.4 CMU output signals to FCCU and INTC

The output signals of the CMUs are connected to the FCCU in the following way:

- The OLR output of CMU_0 is connected to the CMU_0_OSC channel of the FCCU.
- The FLC, FHH, and FLL outputs of CMU_0 are combined (ORed) together and then connected to the CMU_0_PLL channel of the FCCU.
- The FLC, FHH, and FLL outputs of CMU_1 to CMU_3 are all combined (ORed) together and then connected to the CMU_COMP channel of the FCCU.
- The FLC, FHH, and FLL outputs of CMU_4 and the FHH and FLL outputs of CMU_5 to CMU_8 are all combined (ORed) together and then connected to the CMU_OTHER channel of the FCCU.

The output signals of the CMUs are also connected to the INTC in the following way:

- All outputs from CMU_0 and CMU_1 are combined (ORed) together and then connected to interrupt vector 350.

- All outputs from CMU_2 and CMU_3 are combined (ORed) together and then connected to interrupt vector 351.
- All outputs from CMU_4 and CMU_5 are combined (ORed) together and then connected to interrupt vector 352.
- All outputs from CMU_6 and CMU_7 are combined (ORed) together and then connected to interrupt vector 353.
- All outputs from CMU_8 are combined (ORed) together and then connected to interrupt vector 354.

19.13.5 CMU register and field availability

[Table 19-7](#) specifies which registers and fields are available on a given CMU for this device.

Table 19-7. CMU register and field availability

Address offset	Register	CMU	Note
0x0000	CMU_CSR	All	The SFM, CKSEL1, and RCDIV fields are in CMU_0_CSR only.
0x0004	CMU_FDR	CMU_0	—
0x0008	CMU_HFREFR	All	—
0x000C	CMU_LFREFR	All	—
0x0010	CMU_ISR	All	The OLRI field is in CMU_0_ISR only. The FLCI field is available in ISR for CMU_0 to CMU_5 and not available in ISR CMU_6 to CMU_8. The FHFI and FLFI fields are available in ISR for all CMUs.
0x0014	Reserved	—	—
0x0018	CMU_MDR	CMU_0	—

19.13.6 CMU_0_CSR reset value

The chip-specific reset value of CMU_0_CSR is 0000_0000h.

The reset value of CSR for all other CMU instances, and the reset value of other registers for all CMU instances, appear in the CMU's [Register description and memory map](#).

19.13.7 CMU use restrictions

The following restrictions apply to CMU use:

- In CMU_0, $f_{\text{CLKMN0\_RMT}}$ must be greater than $f_{\text{CLKMT0\_RMN}} / 2^{\text{RCDIV}}$ by at least 0.5 MHz to guarantee correct CLKMN0_RMT monitoring. In CMU_1 to CMU_5, f_{CLKMN1} must be greater than $f_{\text{CLKMT0\_RMN}} / 4$ by at least 0.5 MHz to guarantee correct CLKMN1 monitoring. The accuracy of CLKMT0_RMN (IRC) is specified in the chip's Data Sheet.
- If the checker core and RCCU are disabled (via a DCF client), then the clock to the checker core lake is halted to save power. Because CMU_1 and CMU_2 are monitoring points in the clock tree inside the checker core lake, these CMUs cannot be used if the checker core is disabled.

19.13.8 Loss-of-clock behavior

Loss-of-clock detection is provided by the FLC output signals of the CMUs; see [Figure 19-4](#) and [Figure 19-5](#). The FLC output signals are combined with other CMU output signals and then routed to the FCCU, as explained in [CMU output signals to FCCU and INTC](#). The FCCU can be programmed to react to CMU failures with a system reset. Because the FCCU is clocked by the IRC, it can generate the reset request even in the absence of the system clock. The reset signal generated by the FCCU overwrites the system clock selection from SIU_SYSDIV[SYSCLKSEL], forcing the selection of the IRC as the clock source. This mechanism guarantees that the reset state machine continues to receive a (running) clock so that it can propagate the reset request generated by the FCCU.

NOTE

CMU_6 to CMU_8 monitor clocks whose frequency can be lower than the IRC frequency divided by 4, so they do not implement the FLC detection. The other signals (FLL, FHH) do not assert when the monitored clock stops. Therefore these CMUs cannot detect a loss-of-clock condition.

19.14 PLL loss of lock interrupt

Always write 1 to both PLL0CR[LOLIE] and PLL1CR[LOLIE] to keep the loss of lock interrupt function enabled.

The main interrupt functionality is determined by the FCCU via fault input NCF[23] for PLL0 and fault input NCF[24] for PLL1.

19.15 Debug mode

The clock generation modules (PLL0, PLL1, XOSC, and IRC) and clock monitoring units (CMUs) do not support entry into debug mode. Therefore, there is no possibility of freezing these modules in debug mode.

Chapter 20

Progressive Clock Switch (PCS)

20.1 Introduction

This chip contains a Progressive Clock Switch (PCS) module that can be used to prevent sudden voltage drops and overshoots when the system clock frequency is switched from one clock source to another.

20.2 Block diagram

The following figure is a block diagram of the PCS system.

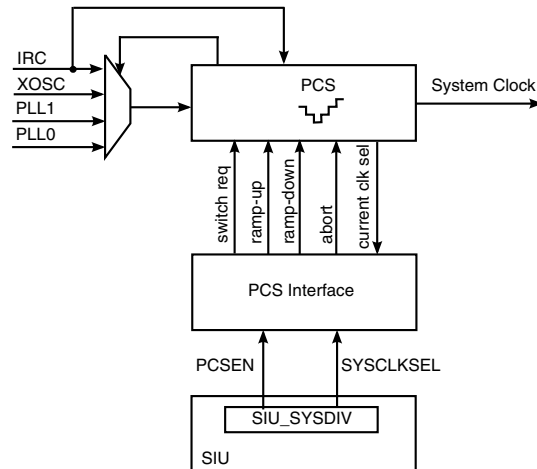


Figure 20-1. PCS system

20.3 Features

The PCS module provides the following features:

- Progressive frequency switching to minimize power supply voltage drops and overshoots
- Clock switching done in two steps: ramp-down from source frequency to IRC frequency, then ramp-up to target frequency
- Fully programmable ramp-down and ramp-up characteristics
- Automatic monitoring of changes on clock switch selector to trigger progressive clock switching feature

20.4 Memory map and register definition

Before programming the PCS registers (PCS_SDUR, PCS_DIVC n , PCS_DIVE n , and PCS_DIVS n), select IRC as the system clock: ensure SIU_SYSDIV[SYSCLKSEL] is 00b. This selection preserves reliable progressive clock operation if, during the PCS register programming, a reset event occurs.

Any access to unused registers and write accesses to read-only registers:

- do not change register content
- cause a transfer error

Unless otherwise noted, all registers may be accessed as 32-bit words, 16-bit half-words, or 8-bit bytes. The byte order is big-endian. For example, the PCS_DIVC1 register's RATE field may be accessed as a word at address offset 704h, as a half-word at address offset 706h, or as a byte at address offset 707h.

PCS memory map

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
700	PCS Switch Duration Register (PCS_SDUR)	32	R/W	0000_0000h	20.4.1/815
704	PCS Divider Change Register for XOSC (PCS_DIVC1)	32	R/W	03E7_0000h	20.4.2/815
708	PCS Divider End Register for XOSC (PCS_DIVE1)	32	R/W	0000_03E7h	20.4.3/816
70C	PCS Divider Start Register for XOSC (PCS_DIVS1)	32	R/W	0000_03E7h	20.4.4/816
710	PCS Divider Change Register for PLL1 (PCS_DIVC2)	32	R/W	03E7_0000h	20.4.5/817
714	PCS Divider End Register for PLL1 (PCS_DIVE2)	32	R/W	0000_03E7h	20.4.6/818
718	PCS Divider Start Register for PLL1 (PCS_DIVS2)	32	R/W	0000_03E7h	20.4.7/818
71C	PCS Divider Change Register for PLL0 (PCS_DIVC3)	32	R/W	03E7_0000h	20.4.8/819
720	PCS Divider End Register for PLL0 (PCS_DIVE3)	32	R/W	0000_03E7h	20.4.9/819
724	PCS Divider Start Register for PLL0 (PCS_DIVS3)	32	R/W	0000_03E7h	20.4.10/820

20.4.1 PCS Switch Duration Register (PCS_SDUR)

This register contains the progressive system clock switching duration for each step.

Address: 0h base + 700h offset = 700h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	SDUR								0																							
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	

PCS_SDUR field descriptions

Field	Description
0–7 SDUR	Switch Duration This value defines the duration of one PCS clock switch step in terms of IRC cycles.
8–31 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

20.4.2 PCS Divider Change Register for XOSC (PCS_DIVC1)

This register defines the rate of frequency change and initial change value for the progressive system clock switching when switching the system clock source to or from the crystal oscillator XOSC on ramp-up and ramp-down, respectively.

NOTE

Byte write accesses are not allowed for the INIT field of this register. Such an access will not result in an exception, but the value will not be loaded with the new value.

Address: 0h base + 704h offset = 704h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	INIT															0						RATE										
W																																
Reset	0	0	0	0	0	0	1	1	1	1	1	0	0	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	

PCS_DIVC1 field descriptions

Field	Description
0–15 INIT	Divider Change Initial Value This is initial change value of the clock divider for the clock ramp-up phase when switching to the XOSC.

Table continues on the next page...

PCS_DIVC1 field descriptions (continued)

Field	Description
16–23 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
24–31 RATE	Divider Change Rate This value controls the change value of the clock divider for the clock ramp-up and ramp-down phase when switching to/from the XOSC.

20.4.3 PCS Divider End Register for XOSC (PCS_DIVE1)

This register defines the final division value for the progressive system clock switching when switching the system clock source from the XOSC on ramp-down.

NOTE

Byte and half-word write accesses are not allowed for this register. Such an access will not result in an exception, but the value will not be loaded with the new value.

Address: 0h base + 708h offset = 708h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0												DIVE																			
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	0	0	1	1	1

PCS_DIVE1 field descriptions

Field	Description
0–11 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
12–31 DIVE	Divider End Value This is the clock divider end value for the clock ramp-down phase when switching from the XOSC.

20.4.4 PCS Divider Start Register for XOSC (PCS_DIVS1)

This register defines the initial division value for the progressive system clock switching when switching the system clock source to the XOSC on ramp-up.

NOTE

Byte and half-word write accesses are not allowed for this register. Such an access will not result in an exception, but the value will not be loaded with the new value.

Address: 0h base + 70Ch offset = 70Ch

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0												DIVS																			
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	0	0	1	1	1

PCS_DIVS1 field descriptions

Field	Description
0–11 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
12–31 DIVS	Divider Start Value This is the start value of the clock divider for the clock ramp-up phase when switching to the XOSC.

20.4.5 PCS Divider Change Register for PLL1 (PCS_DIVC2)

This register defines the rate of frequency change and initial change value for the progressive system clock switching when switching the system clock source to or from PLL1 on ramp-up and ramp-down, respectively.

NOTE

Byte write accesses are not allowed for the INIT field of this register. Such an access will not result in an exception, but the value will not be loaded with the new value.

Address: 0h base + 710h offset = 710h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	INIT															0							RATE									
W																																
Reset	0	0	0	0	0	0	1	1	1	1	1	0	0	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

PCS_DIVC2 field descriptions

Field	Description
0–15 INIT	Divider Change Initial Value This is initial change value of the clock divider for the clock ramp-up phase when switching to PLL1.
16–23 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
24–31 RATE	Divider Change Rate This value controls the change value of the clock divider for the clock ramp-up and ramp-down phase when switching to/from PLL1.

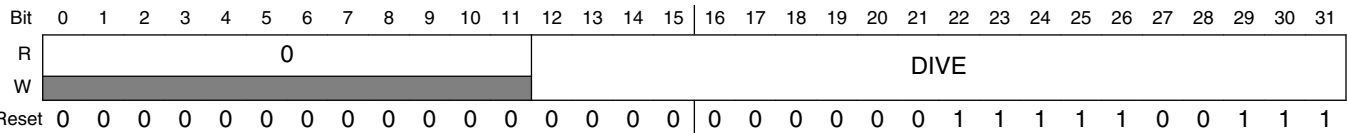
20.4.6 PCS Divider End Register for PLL1 (PCS_DIVE2)

This register defines the final division value for the progressive system clock switching when switching the system clock source from PLL1 on ramp-down.

NOTE

Byte and half-word write accesses are not allowed for this register. Such an access will not result in an exception, however the value will not be loaded with the new value.

Address: 0h base + 714h offset = 714h



PCS_DIVE2 field descriptions

Field	Description
0–11 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
12–31 DIVE	Divider End Value This is the clock divider end value for the clock ramp-down phase when switching from PLL1.

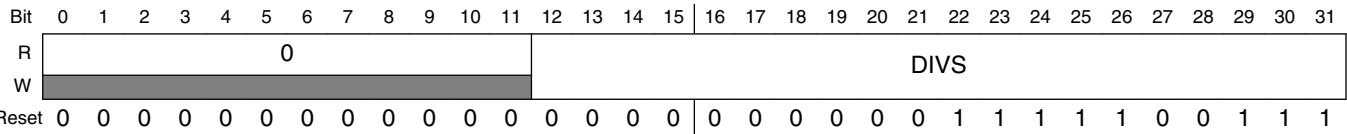
20.4.7 PCS Divider Start Register for PLL1 (PCS_DIVS2)

This register defines the initial division value for the progressive system clock switching when switching the system clock source to PLL1 on ramp-up.

NOTE

Byte and half-word write accesses are not allowed for this register. Such an access will not result in an exception, but the value will not be loaded with the new value.

Address: 0h base + 718h offset = 718h



PCS_DIVS2 field descriptions

Field	Description
0–11 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
12–31 DIVS	Divider Start Value This is the start value of the clock divider for the clock ramp-up phase when switching to PLL1.

20.4.8 PCS Divider Change Register for PLL0 (PCS_DIVC3)

This register defines the rate of frequency change and initial change value for the progressive system clock switching when switching the system clock source to or from PLL0 on ramp-up and ramp-down, respectively.

NOTE

Byte write accesses are not allowed for the INIT field of this register. Such an access will not result in an exception, but the value will not be loaded with the new value.

Address: 0h base + 71Ch offset = 71Ch

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R																																
W																																
Reset	0	0	0	0	0	0	1	1	1	1	1	0	0	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

PCS_DIVC3 field descriptions

Field	Description
0–15 INIT	Divider Change Initial Value This is initial change value of the clock divider for the clock ramp-up phase when switching to PLL0.
16–23 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
24–31 RATE	Divider Change Rate This value controls the change value of the clock divider for the clock ramp-up and ramp-down phase when switching to/from PLL0.

20.4.9 PCS Divider End Register for PLL0 (PCS_DIVE3)

This register defines the final division value for the progressive system clock switching when switching the system clock source from PLL0 on ramp-down.

NOTE

Byte and half-word write accesses are not allowed for this register. Such an access will not result in an exception, but the value will not be loaded with the new value.

Address: 0h base + 720h offset = 720h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0												DIVE																			
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	0	0	1	1	1

PCS_DIVE3 field descriptions

Field	Description
0–11 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
12–31 DIVE	Divider End Value This is the clock divider end value for the clock ramp-down phase when switching from PLL0.

20.4.10 PCS Divider Start Register for PLL0 (PCS_DIVS3)

This register defines the initial division value for the progressive system clock switching when switching the system clock source to PLL0 on ramp-up.

NOTE

Byte and half-word write accesses are not allowed for this register. Such an access will not result in an exception, but the value will not be loaded with the new value.

Address: 0h base + 724h offset = 724h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0												DIVS																			
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	0	0	1	1	1

PCS_DIVS3 field descriptions

Field	Description
0–11 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
12–31 DIVS	Divider Start Value This is the start value of the clock divider for the clock ramp-up phase when switching to PLL0.

20.5 Functional description

The progressive clock switch feature is enabled by setting SIU_SYSDIV[PCSEN]. If PCSEN is set and a change to SIU_SYSDIV[SYSCLKSEL] is detected by the PCS interface logic (see the PCS block diagram), then a progressive clock switch operation is triggered in the PCS block. The following rules govern the operation:

- If the switching is from the IRC clock, then a ramp-up command is sent to the PCS.
- If switching is to the IRC clock, then a ramp-down command is sent to the PCS.
- If the switching is between two clock sources and neither one is the IRC, then ramp-down and ramp-up commands are simultaneously sent to the PCS.

When the PCS block receives ramp-up and a ramp-down commands simultaneously, it executes the processes in this sequence:

1. Ramp-down process
2. Ramp-up process

During ramp-down, the rate of the frequency change is based on the PCS_SDUR, PCS_DIVC n , and PCS_DIVE n registers, where n corresponds to the current system clock source selection. During ramp-up, the rate of the frequency change is based on the PCS_SDUR, PCS_DIVC n , and PCS_DIVS n registers, where n corresponds to the target system clock source selection.

NOTE

The progressive clock switching feature assumes that the IRC has the lowest clock frequency of the device. Do not attempt to use PCS when switching to or from a frequency that is less than than the IRC frequency.

When the PCS feature is enabled, the PCS module itself—not the SIU_SYSDIV[SYSCLKSEL] field—controls the selection of the clock source.

While a PCS switching process is in progress:

- The clock source selected by the PCS may differ from the one requested by SIU_SYSDIV[SYSCLKSEL].

- If software writes a new value to SIU_SYSDIV[SYSCLKSEL] that corresponds to a clock source different from the one at the start of the switching process, the new value is ignored until the PCS process finishes.
- If software writes a SIU_SYSDIV[SYSCLKSEL] value corresponding to the same clock source that was in effect at the start of the switching process, the PCS process aborts and returns to that original clock source.

At the end of the clock switching process:

- The SIU_PCSIFR[PCSI] flag is asserted in the SIU. An interrupt to the CPU is generated if enabled by SIU_PCSE[PCSE].
- The SIU_PCSIFR[PCSMS] field indicates the current clock source selected by the PCS. Software can read PCSMS to learn whether or not the PCS switching process succeeded.

If the PCS switching fails (for example, due to incorrect programming), the clock source will not switch to a different one requested by SIU_SYSDIV[SYSCLKSEL]. Instead, the clock source remains the one selected at the beginning of the attempted PCS switch.

20.5.1 Generic clock change requirements

For a maximum allowed change of the frequency (f_{chg}) and for a given source clock frequency f_{src} (current or target clock source), the following equation gives the maximum allowed frequency change rate a_{max} .

$$a_{\text{max}} = \frac{f_{\text{chg}}}{f_{\text{src}}}$$

20.5.2 Configuration of PCS_SDUR

The switch duration field PCS_SDUR[SDUR] defines the duration of one system clock switch step in terms of the IRC clock period. After the expiration of this time, the module changes the clock divider value, which changes the frequency of the system clock.

20.5.3 Configuration of PCS_DIVCn[RATE]

For a given maximum allowed frequency change rate a_{max} , the following table shows the value d based on the value programmed into the PCS_DIVCn[RATE] register.

Table 20-12. PCS_DIVCn[RATE] values

$a_{\max}$	d	PCS_DIVCn[RATE]
0.05	0.012	12
0.10	0.048	48
0.15	0.112	112
0.20	0.184	184

20.5.4 Generic clock change properties

The following equation gives the number k of steps required to ramp down the source frequency (f_{src}) to reach the IRC frequency (f_{irc}).

$$k = \left\lceil 0.5 + \sqrt{0.25 - \frac{2 \left(1 - \frac{f_{\text{src}}}{f_{\text{irc}}} \right)}{d}} \right\rceil$$

The following equation gives the number k of steps required to ramp up the IRC frequency (f_{irc}) to reach the target frequency (f_{targ}).

$$k = \left\lceil 0.5 + \sqrt{0.25 - \frac{2 \left(1 - \frac{f_{\text{targ}}}{f_{\text{irc}}} \right)}{d}} \right\rceil$$

20.5.5 Clock ramp-down

The clock ramp-down:

- starts with the divider value 1 and with the given divider increment value PCS_DIVCn[RATE].
- ends with the given divider value PCS_DIVE n [DIVE].

The following equation gives the divider end value for clock ramp-down.

$$\text{PCS\_DIVE}n[\text{DIVE}] = \frac{f_{\text{curr}}}{f_{\text{irc}}} \times 1000 - 1$$

In the equation, f_{curr} is the frequency of the currently selected system clock source, and f_{irc} is the frequency of the IRC clock.

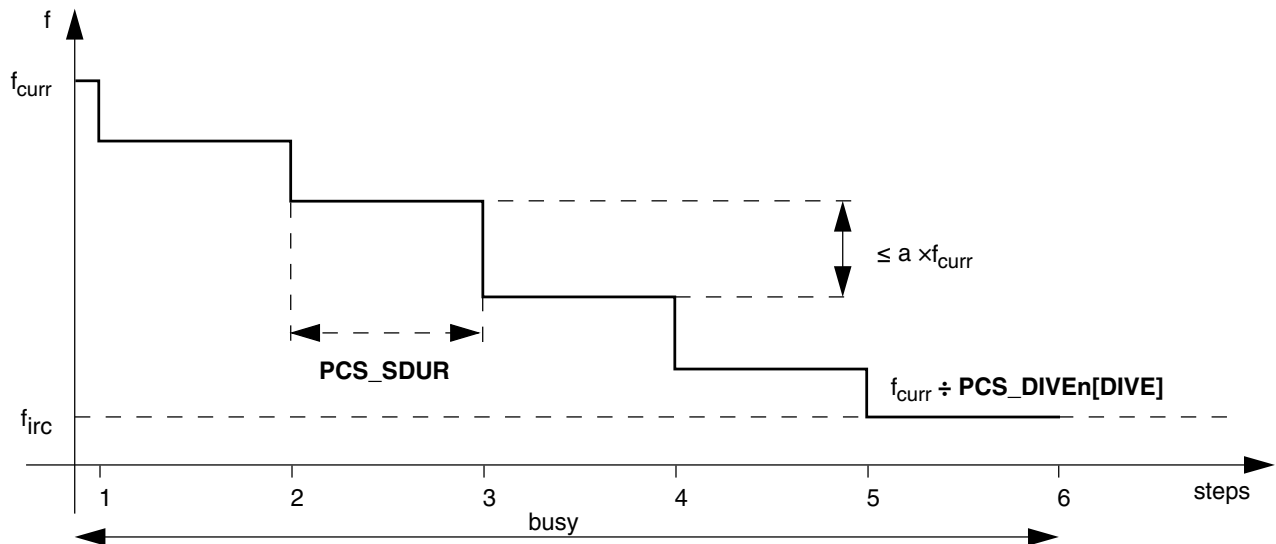


Figure 20-12. System clock ramp-down timing (k = 6 example)

20.5.6 Clock ramp-up

The clock ramp-up:

- starts with the given divider value $PCS\_DIVS_n[DIVS]$ and with the given divider decrement value $PCS\_DIVE_n[INIT]$.
- ends with the divider value 1.

The following equation gives the initial divider change start value $PCS\_DIVE_n[INIT]$ for system clock ramp-up.

$$PCS\_DIVE_n[INIT] = d \times k \times 1000$$

In the preceding equation, k is calculated from the second equation in [Generic clock change properties](#). The following equation gives the divider start value for clock ramp-up.

$$PCS\_DIVS_n[DIVS] = \left(1 + d^{\frac{k(k+1)}{2}}\right) \times 1000 - 1$$

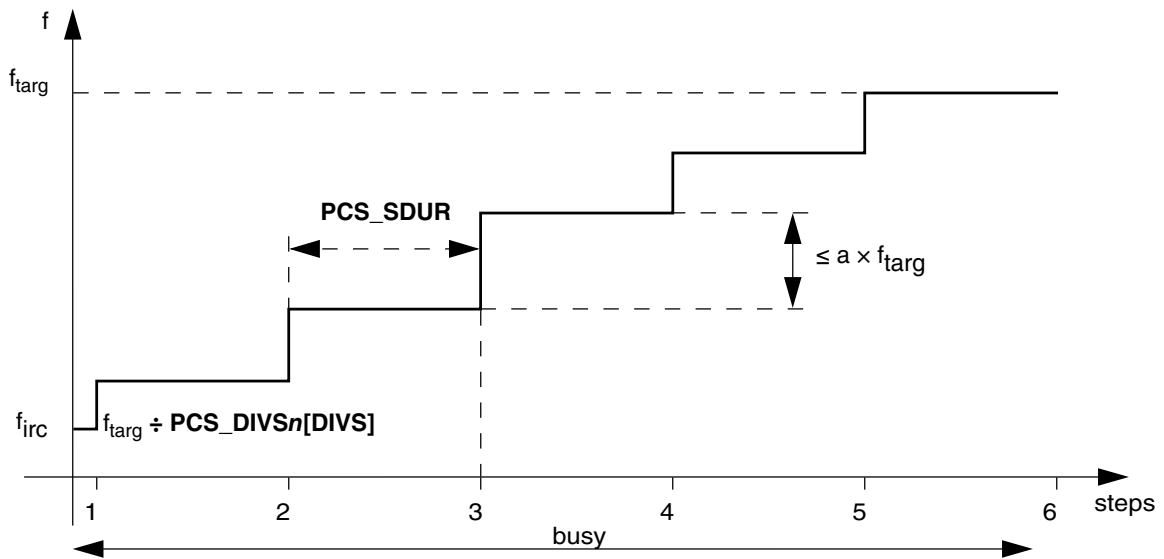


Figure 20-13. System clock ramp-up timing (k = 6 example)

Chapter 21

Dual PLL Digital Interface (PLLDIG)

21.1 Chip-specific PLLDIG information

The chip has one instance of the PLLDIG module. It includes two PLLs: PLL0 and PLL1.

For more chip-specific information, see the [Clocking](#) details, especially the [PLL](#) details.

21.2 Introduction

The chip provides a user interface and control over the dual PLL system composed of PLL0 and PLL1 analog blocks and the digital interface (PLLDIG). The two analog PLL blocks are cascaded, with the PHI1 output of PLL0 feeding the clock input of PLL1.

A key feature of the dual PLL architecture is the ability to drive peripherals from the PLL0 PHI output, which is non-modulated and independent of the core clock frequency. The core and platform clocks are driven by the PLL1 PHI frequency-modulated output.

21.3 Acronyms and Abbreviations

The following table contains sample acronyms and abbreviations used in this chapter.

Table 21-1. Acronyms and abbreviated terms

Term	Meaning
PLL	Phase Lock Loop
RFD	Reduced Frequency Divider
MFD	Multiplication Factor Divider
VCO	Voltage Controlled Oscillator
FMPLL	Frequency Modulated Phase Locked Loop

21.4 Block Diagram

Please refer to the “Clocking chapter” of this *Reference Manual* to see the block diagram.

21.5 Features

The Dual PLLDIG has the following features:

- Supports dual PLL in cascaded mode with PLL0 clock out as reference clock to PLL1.
- Reference clock pre-divider for finer frequency synthesis resolution.
- Reduced frequency divider for decreasing the PLL0/PLL1 output clock frequency without causing the PLLs to lose lock.
- Programmable frequency modulation on PLL1.
- Lock detect circuitry reports when the PLLs have achieved frequency lock, and continuously monitors lock status to report loss of lock conditions.
 - User-selectable ability to generate an interrupt request upon loss of lock.
 - The loss of lock indication is sent to the FCCU via the system glue logic.

21.6 Modes of operation

The operating mode of the PLLs is indicated by the value of PLL n CR[CLKCFG]. The various modes of operation for the PLL are:

- Normal mode with reference, with either PLL0 or both PLLs enabled
- PLL disabled

Normal mode is defined as the mode where the clocks are driven by the PLLs. When using a crystal for the clock reference, reset should remain asserted until the oscillator has stabilized.

21.6.1 Normal mode with reference, PLL0 or both PLLs enabled

In normal mode, PLL0 receives an input clock from the reference that can be prescaled by the pre-divider. PLL0 multiplies the frequency to create the PLL0 output clock. The user must supply a crystal that is within the appropriate frequency range, the crystal manufacturer's recommended external support circuitry and short signal route from the MCU to the crystal.

For information about the selection of a clock reference for each PLL, see [PLL](#) in the Clocking details.

PLL0 generates a non-modulated clock which drives PLL0 PHI1. PLL1 can generate a frequency modulated clock or a non-modulated clock (for example, locked on a single frequency). Write the PLL1FM register to program the modulation rate, modulation depth, and whether PLL modulating is enabled.

NOTE

PLL1 must be configured to work with a reference clock from PLL0 only after PLL0 has achieved lock.

For proper power sequencing of analog PLL blocks, write to PLL0CR[CLKCFG] before writing to PLL1CR[CLKCFG].

While powering down the PLLs: If PLL0:PHI1 is used as the source for PLL1, power down the PLLs in this sequence to ensure proper shutoff:

1. PLL1
2. PLL0

21.7 Memory map and register definition

This section provides the memory map and detailed descriptions of all registers for configuring the PLLs. The table below shows the memory map. Addresses are given as offsets from the module base address. All registers can be accessed using 8-bit, 16-bit or 32-bit addressing.

PLLDIG memory map

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/page
0	PLLDIG PLL0 Control Register (PLLDIG_PLL0CR)	32	R/W	0000_0000h	21.7.1/830

Table continues on the next page...

PLLDIG memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
4	PLLDIG PLL0 Status Register (PLLDIG_PLL0SR)	32	R/W	0000_0000h	21.7.2/831
8	PLLDIG PLL0 Divider Register (PLLDIG_PLL0DV)	32	R/W	2001_1008h	21.7.3/832
20	PLLDIG PLL1 Control Register (PLLDIG_PLL1CR)	32	R/W	0000_0000h	21.7.4/834
24	PLLDIG PLL1 Status Register (PLLDIG_PLL1SR)	32	R/W	0000_0000h	21.7.5/835
28	PLLDIG PLL1 Divider Register (PLLDIG_PLL1DV)	32	R/W	0001_0010h	21.7.6/836
2C	PLLDIG PLL1 Frequency Modulation Register (PLLDIG_PLL1FM)	32	R/W	0000_0000h	21.7.7/837
30	PLLDIG PLL1 Fractional Divide Register (PLLDIG_PLL1FD)	32	R/W	0000_0000h	21.7.8/838

21.7.1 PLLDIG PLL0 Control Register (PLLDIG_PLL0CR)

Address: 0h base + 0h offset = 0h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0						CLKCFG		0				LOLIE	0		
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

PLLDIG_PLL0CR field descriptions

Field	Description
0–21 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
22–23 CLKCFG	Clock Configuration This field can be used to change the operating mode. 00 PLL off 01 Reserved 10 Reserved 11 Normal mode with PLL running
24–27 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
28 LOLIE	Loss-of-lock interrupt enable

Table continues on the next page...

PLLDIG_PLL0CR field descriptions (continued)

Field	Description
	The LOLIE bit enables a loss-of-lock interrupt request when PLL0SR[LOLF] = 1. If PLL0SR[LOLF] = 0 or PLL0CR[LOLIE] = 0, the interrupt request is ignored. The interrupt request occurs only in normal mode, so the LOLIE bit has no effect if PLL is disabled. NOTE: See PLL loss of lock interrupt in the Clocking details for more information about configuring this field. 0 Ignore loss-of-lock. Interrupt not requested. 1 Enable interrupt request upon loss-of-lock.
29–31 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

21.7.2 PLLDIG PLL0 Status Register (PLLDIG_PLL0SR)

Address: 0h base + 4h offset = 4h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0												LOLF	LOCK	0	
W													w1c			
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

PLLDIG_PLL0SR field descriptions

Field	Description
0–27 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
28 LOLF	Loss-of-lock flag

Table continues on the next page...

PLLDIG_PLL0SR field descriptions (continued)

Field	Description
	This bit provides the interrupt request flag for the loss-of-lock condition. Software must write 1 to LOLF to clear it to 0; writing 0 has no effect. The flag is set to 1 when the PLL loses lock or when the divider/modulation registers are changed on the fly. This flag bit is sticky in the sense that if lock is reacquired, the bit remains set to 1 until cleared by either writing 1 or asserting reset. The LOLF is not asserted when the PLL is disabled, but disabling the PLL will not automatically clear the LOLF if it was previously asserted while the PLL was in normal mode. 0 No loss of lock detected. Interrupt service not requested. 1 Loss of lock detected. Interrupt service requested.
29 LOCK	Lock status bit Indicates whether PLL has acquired lock. 0 PLL is unlocked. 1 PLL is locked.
30–31 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

21.7.3 PLLDIG_PLL0 Divider Register (PLLDIG_PLL0DV)

The PLL0DV register provides the PHI/PHI1 output clock reduced frequency divider (RFDPHI and RFDPHI1), pre-divider (PREDIV), and loop divider (MFD). If this register's fields are modified without powering down the PLL, the PLL will lose lock and generate either an interrupt or a failure signal to the FCCU, based on which is enabled.

Address: 0h base + 8h offset = 8h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0							0								
W																
Reset	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	1

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0							0								
W																
Reset	0	0	0	1	0	0	0	0	0	0	0	0	1	0	0	0

PLLDIG_PLL0DV field descriptions

Field	Description
0 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
1–4 RFDPHI1	PHI1 reduced frequency divider This 4-bit field determines the post-divider value for driving the PHI1 output clock.

Table continues on the next page...

PLLDIG_PLL0DV field descriptions (continued)

Field	Description
	0h to 3h Reserved 4h Divide by 4 5h Divide by 5 Eh Divide by 14 Fh Divide by 15
5–9 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
10–15 RFDPHI	PHI reduced frequency divider This 6-bit field determines the VCO clock post-divider for driving the PHI output clock. See the equations in Clock configuration .
16 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
17–19 PREDIV	Input clock predivider This 3-bit field controls the value of the divider on the input clock. The output of the predivider circuit generates the reference clock to the PLL analog loop. The PREDIV value 000b causes the input clock to be inhibited. 000 Clock inhibit 001 Divide by 1 010 Divide by 2 011 Divide by 3 100 Divide by 4 101 Divide by 5 110 Divide by 6 111 Divide by 7
20–24 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
25–31 MFD	Loop multiplication factor divider This 7-bit field controls the value of the divider in the feedback loop. The value specified by the MFD bits establishes the multiplication factor applied to the reference frequency. 00h to 07h Reserved 08h Divide by 8 09h Divide by 9 7Eh Divide by 126 7Fh Divide by 127

21.7.4 PLLDIG PLL1 Control Register (PLLDIG_PLL1CR)

Address: 0h base + 20h offset = 20h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0						CLKCFG		0				LOLIE	0		
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

PLLDIG_PLL1CR field descriptions

Field	Description
0–21 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
22–23 CLKCFG	Clock Configuration This field can be used to change the operating mode. 00 PLL off 01 Reserved 10 Reserved 11 Normal mode with PLL running
24–27 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
28 LOLIE	Loss-of-lock interrupt enable The LOLIE bit enables a loss-of-lock interrupt request when PLL1SR[LOLF] = 1. If PLL1SR[LOLF] = 0 or PLL1CR[LOLIE] = 0, the interrupt request is ignored. The interrupt request occurs only in normal mode, so the LOLIE bit has no effect if PLL is disabled. NOTE: See PLL loss of lock interrupt in the Clocking details for more information about configuring this field. 0 Ignore loss-of-lock. Interrupt not requested. 1 Enable interrupt request upon loss-of-lock.
29–31 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

21.7.5 PLLDIG PLL1 Status Register (PLLDIG_PLL1SR)

Address: 0h base + 24h offset = 24h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15														
R	0																													
W																														
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0														
Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31														
R	0																											LOLF	LOCK	0
W																												w1c		
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0														

PLLDIG_PLL1SR field descriptions

Field	Description
0–27 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
28 LOLF	Loss-of-lock flag This bit provides the interrupt request flag for the loss-of-lock condition. Software must write 1 to LOLF to clear it to 0; writing 0 has no effect. The flag is set to 1 when the PLL loses lock or when the divider/modulation registers are changed on the fly. This flag bit is sticky in the sense that if lock is reacquired, the bit remains set to 1 until cleared by either writing 1 or asserting reset. The LOLF is not asserted when the PLL is disabled, but disabling the PLL will not automatically clear the LOLF if it was previously asserted while the PLL was in normal mode. 0 No loss of lock detected. Interrupt service not requested. 1 Loss of lock detected. Interrupt service requested.
29 LOCK	Lock status bit Indicates whether PLL has acquired lock. 0 PLL is unlocked. 1 PLL is locked.
30–31 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

21.7.6 PLLDIG PLL1 Divider Register (PLLDIG_PLL1DV)

The PLL1DV register provides the PHI output clock reduced frequency divider (RFDPHI) and loop divider (MFD). If this register's fields are modified without powering down the PLL, the PLL will lose lock and generate either an interrupt or a failure signal to the FCCU, based on which is enabled.

Address: 0h base + 28h offset = 28h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0										RFDPHI						0									MFD						
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0

PLLDIG_PLL1DV field descriptions

Field	Description
0–9 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
10–15 RFDPHI	PHI reduced frequency divider This 6-bit field determines the VCO clock post-divider for driving the PHI output clock. See the equations in Clock configuration .
16–24 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
25–31 MFD	Loop multiplication factor divider This 7-bit field controls the value of the divider in the feedback loop. The value specified by the MFD bits establishes the multiplication factor applied to the reference frequency. 00h to 09h Reserved 10h Divide by 16 11h Divide by 17 21h Divide by 33 22h Divide by 34 23h to 7Fh Reserved

21.7.7 PLLDIG PLL1 Frequency Modulation Register (PLLDIG_PLL1FM)

The PLL1FM register enables frequency modulation on PLL1 and provides controls for the FM spread spectrum, modulation depth, and modulation rate. This register should be written when PLL1CR[CLKCFG] = 00b (PLL1 powered down). After the PLL is on and has locked, changing the values of PLL1FM[MODEN] and PLL1FM[MODSEL] can result in the PLL losing its lock. The PLL would then require power cycling to regain normal functionality.

Address: 0h base + 2Ch offset = 2Ch

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	MODEN	MODSEL	MODPRD												
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	INCSTP														
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

PLLDIG_PLL1FM field descriptions

Field	Description
0 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
1 MODEN	Modulation enable This bit enables the frequency modulation. 0 Frequency modulation disabled. 1 Frequency modulation enabled.
2 MODSEL	Modulation selection This bit selects whether modulation is centered around the nominal frequency or spread below the nominal frequency. 0 Center Spread modulation: Modulation centered around nominal frequency. 1 Down Spread modulation: Modulation spread below nominal frequency.
3–15 MODPRD	Modulation period

Table continues on the next page...

PLLDIG_PLL1FM field descriptions (continued)

Field	Description
	$\text{MODPRD} = \text{round}\left(\frac{f_{\text{ref}}}{4 \times f_{\text{mod}}}\right)$ MODPRD is the modulation period variable derived from the formula: where f_{ref} represents the frequency of the feedback divider and f_{mod} represents the modulation frequency.
16 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
17–31 INCSTP	Increment step This field is the INCSTP variable derived from the formula: $\text{INCSTEP} = \text{round}\left(\frac{(2^{15} - 1) \times md \times \text{PLL1DV}[\text{MFD}]}{100 \times 5 \times \text{MODPRD}}\right)$ where md represents the peak modulation depth in percentage (+/- for centered modulation, $-2 \times md$ for modulation below nominal frequency) and MFD represents the nominal value of the feedback loop divider.

21.7.8 PLLDIG PLL1 Fractional Divide Register (PLLDIG_PLL1FD)

The PLL1FD register provides the enable and fractional divide factor for the loop divider. This register should be written when PLL1CR[CLKCFG] = 00b (PLL1 powered down). After the PLL is running and has locked, changing the value of FDEN can result in the PLL losing its lock. The PLL would then require power cycling to regain normal functionality.

Address: 0h base + 30h offset = 30h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0								0							
W		FDEN													DTHDIS1	DTHDIS0
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R			0													
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

PLLDIG_PLL1FD field descriptions

Field	Description
0 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

Table continues on the next page...

PLLDIG_PLL1FD field descriptions (continued)

Field	Description
1 FDEN	Fractional Divide Enable This bit enables the fractional divider in the loop divider for PLL1. 0 Fractional divide disabled. 1 Fractional divide enabled.
2–13 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
14 DTHDIS1	Rectangular Dither Disable Disables the Rectangular Phase/Frequency Detector dither. 0 Dither enabled 1 Dither disabled
15 DTHDIS0	Triangular Dither Disable Disables the Triangular Phase/Frequency Detector dither. 0 Dither enabled 1 Dither disabled
16–19 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
20–31 FRCDIV	Fractional divide input When the fractional divide is disabled, the VCO clock frequency is the product of the input clock and the loop divide factor (MFD). When fractional divide is enabled and dither is disabled (DTHDIS1 is 1), the VCO clock frequency is given by: $\text{VCO Frequency} = \text{InputClock} \times (\text{PLL1DV}[\text{MFD}] + (\text{FRCDIV} \div 2^{12}))$ When fractional divide is enabled and dither is enabled (DTHDIS1 is 0), the VCO clock frequency is given by: $\text{VCO Frequency} = \text{InputClock} \times (\text{PLL1DV}[\text{MFD}] + (\text{FRCDIV} \div 2^{12}) + (1 \div 2^{13}))$ In these equations, FRCDIV is the decimal equivalent of the FRCDIV field.

21.8 Functional description

This section explains the operation and configuration of the Dual PLLDIG module.

21.8.1 Input clock frequency

PLL0 and PLL1 are designed to operate over an input clock frequency range. The operating ranges for the PLLs are discussed in detail in the *Data Sheet*.

21.8.2 Clock configuration

The relationship between input and output frequency is determined by programming the PLL0DV and PLL1DV registers and calculated according to the following equations.

NOTE

In the following equations, use the programmed value for each register field. For example, when PLL0DV[RFDPHI] is programmed to the value 1, use the value 1 in the equation for that field.

$$f_{\text{pll0\_phi}} = f_{\text{pll0\_ref}} \times \frac{\text{PLL0DV[MFD]}}{\text{PLL0DV[PREDIV]} \times \text{PLL0DV[RFDPHI]}}$$

Figure 21-9. PLL0 PHI output frequency

$$f_{\text{pll0\_phi1}} = f_{\text{pll0\_ref}} \times \frac{\text{PLL0DV[MFD]}}{\text{PLL0DV[PREDIV]} \times \text{PLL0DV[RFDPHI1]}}$$

Figure 21-10. PLL0 PHI1 output frequency

$$f_{\text{pll1\_phi}} = f_{\text{pll1\_ref}} \times \frac{\text{PLL1DV[MFD]}}{2 \times \text{PLL1DV[RFDPHI]}}$$

Figure 21-11. PLL1 PHI output frequency (PLLDIG_PLL1FD[FDEN] = 0)

$$f_{\text{pll1\_phi}} = f_{\text{pll1\_ref}} \times \left(\frac{\text{PLL1DV[MFD]} + \frac{\text{PLL1FD[FRCDIV]}}{2^{12}} + \frac{1}{2^{13}}}{2 \times \text{PLL1DV[RFDPHI]}} \right)$$

Figure 21-12. PLL1 PHI output frequency (PLLDIG_PLL1FD[FDEN] = 1 and PLLDIG_PLL1FD[DTHDIS1] = 0)

NOTE

When PLLDIG_PLL1FD[FDEN] = 1 and PLLDIG_PLL1FD[DTHDIS1] = 1, in [Figure 21-12](#) remove from the numerator: + 1/(2¹³)

$$f_{\text{pll1\_phi}} = \frac{f_{\text{pll0\_ref}} \times \text{PLL0DV}[\text{MFD}] \times \text{PLL1DV}[\text{MFD}]}{\text{PLL0DV}[\text{PREDIV}] \times \text{PLL0DV}[\text{RFDPHI1}] \times \text{PLL1DV}[\text{RFDPHI}] \times 2}$$

Figure 21-13. PLL1 PHI output frequency when PLL0 PHI1 is PLL1 reference (PLLDIG_PLL1FD[FDEN] = 0)

$$f_{\text{pll1\_phi}} = \frac{f_{\text{pll0\_ref}} \times \text{PLL0DV}[\text{MFD}] \times (\text{PLL1DV}[\text{MFD}] + \frac{\text{PLL1FD}[\text{FRCDIV}]}{2^{12}} + \frac{1}{2^{13}})}{\text{PLL0DV}[\text{PREDIV}] \times \text{PLL0DV}[\text{RFDPHI1}] \times \text{PLL1DV}[\text{RFDPHI}] \times 2}$$

Figure 21-14. PLL1 PHI output frequency when PLL0 PHI1 is PLL1 reference (PLLDIG_PLL1FD[FDEN] = 1 and PLLDIG_PLL1FD[DTHDIS1] = 0)

NOTE

When PLLDIG_PLL1FD[FDEN] = 1 and PLLDIG_PLL1FD[DTHDIS1] = 1, in [Figure 21-14](#) remove from the numerator: + 1/(2¹³)

The relationship between the VCO frequency (f_{VCO}) and the output frequency of the PLLs is determined by the value of PLL1DV[RFDPHI], PLL0DV[RFDPHI], and PLL0DV[RFDPHI1], according to the following equations:

$$f_{\text{pll0\_VCO}} = f_{\text{pll0\_phi}} \times 2 \times \text{PLL0DV}[\text{RFDPHI}]$$

$$f_{\text{pll1\_VCO}} = f_{\text{pll1\_phi}} \times 2 \times \text{PLL1DV}[\text{RFDPHI}]$$

$$f_{\text{pll0\_VCO}} = f_{\text{pll0\_phi1}} \times 2 \times \text{PLL0DV}[\text{RFDPHI1}]$$

NOTE

$f_{\text{pll0\_phi1}}$ is the reference clock generated by PLL0 for PLL1.

When programming the PLLs, user software must not violate the maximum system clock frequency or max/min VCO frequency specification PLL0 and PLL1. Furthermore, the PLL0DV[PREDIV] value must not be set to any value that causes the input frequency to the phase detector of analog PLL blocks to go below the prescribed ranges.

The PLL n SR[LOCK] flag is immediately negated if the fields of PLL n DV are changed without powering down the analog PLLs.

NOTE

The PLL must be powered down and powered up by writing 00b to the PLL n CR[CLKCFG] fields before PLL0DV[PREDIV], PLL0DV[MFD], PLL1DV[MFD], or the reference input clock values are modified.

The recommended procedure to program the PLLs and engage normal mode appears in [Initialization information](#).

21.8.3 Loss of lock

In the event that the PLL loses lock, the PLL digital interface registers provide the flexibility to select whether to generate an interrupt. If an interrupt is asserted, a failure event is also sent to the FCCU.

A loss-of-lock interrupt and FCCU failure event are generated only when the PLL is operating in normal mode. The PLL n SR[LOLF] bits are not asserted when the PLLs are disabled (PLL n CR[CLKCFG]=00b). If an LOLF flag was asserted while the PLL was in normal mode and then the PLL is disabled, the PLL disabling does not automatically clear the flag.

The lock indication from the analog PLL is synchronized and stored in the status register. When the analog PLL loses lock or regains lock, the status register indicates it immediately (after synchronization time).

21.8.4 Frequency modulation

Frequency modulation uses a triangular profile as shown in [Figure 21-15](#). The modulation frequency and depth are controlled using PLL1FM[MODPRD] and PLL1FM[INCSTP].

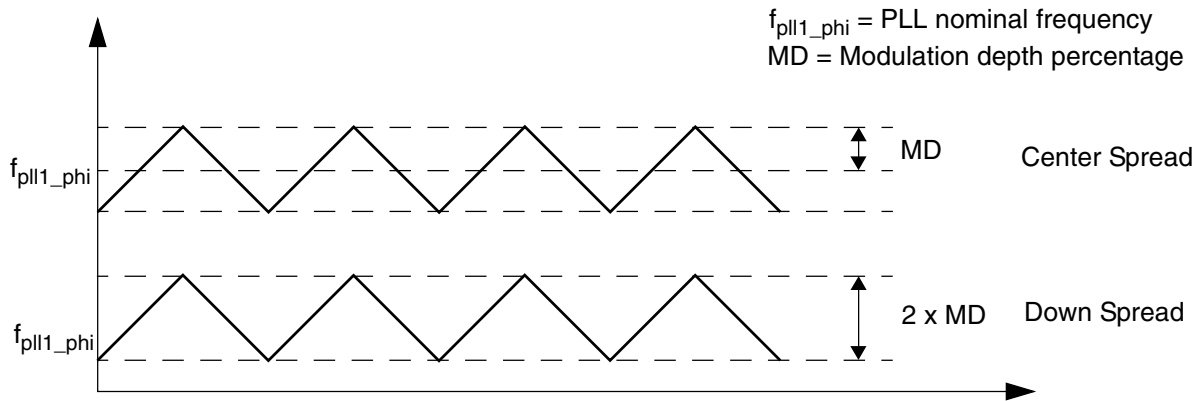


Figure 21-15. Triangular frequency modulation

NOTE

The device maximum operating frequency includes the frequency modulation. If center modulation is used, the f_{sys} must be below f_{max} by MD percentage such that $f_{\text{max}} = f_{\text{sys}} \times (1 + (\text{MD}\% / 100))$.

The following equations define how to calculate PLL1FM[MODPRD] and PLL1FM[INCSTP] based on the frequency of the feedback divider (f_{ref}), the modulation frequency (f_{mod}) and the modulation depth percentage (MD).

$$\text{PLL1FM}[\text{MODPRD}] = \text{round} \left(\frac{f_{\text{pll1\_ref}}}{4 \times f_{\text{mod}}} \right)$$

Figure 21-16. PLL1FM[MODPRD] calculation

The equation to determine PLLFM[INCSTP] is shown in [Figure 21-17](#).

$$\text{PLL1FM}[\text{INCSTP}] = \text{round} \left(\frac{(2^{15} - 1) \times \text{MD} \times \text{PLL1DV}[\text{MFD}]}{100 \times 5 \times \text{PLL1FM}[\text{MODPRD}]} \right)$$

Figure 21-17. PLL1FM[INCSTP] calculation

PLL1FM[MODPRD] and PLL1FM[INCSTP] are subject to the following restriction:

$$(\text{PLL1FM}[\text{MODPRD}] \times \text{PLL1FM}[\text{INCSTP}]) < 2^{15}$$

Figure 21-18. PLL1FM[MODPRD] and PLL1FM[INCSTP] restriction

Due to the preceding rounding operations, the effective modulation depth applied to the FMPLL is as shown in [Figure 21-19](#).

$$\text{ModulationDepth} = \text{round} \left(\frac{\text{PLL1FM}[\text{MODPRD}] \times \text{PLL1FM}[\text{INCSTP}] \times 100 \times 5}{(2^{15} - 1) \times \text{PLL1DV}[\text{MFD}]} \right)$$

Figure 21-19. Modulation Depth

As an example, suppose the following configuration, in which PLL1DV[RFDPHI] represents the programmed value of that field:

- Input frequency: 40 MHz
- Loop divider (PLL1DV[MFD]): 20
- VCO frequency: 40 MHz × 20 = 800 MHz
- PLL output frequency: 800 MHz / (2 × PLL1DV[RFDPHI])
- Center spread (PLL1FM[MODSEL] = 0)
- Modulation frequency: 100 kHz
- Modulation depth: +/- 2.0% (4% peak-to-peak)
- PLL1FM[MODPRD] = round[(40 × 10⁶) / (4 × 100 × 10³)] = round[100.00] = 100
- PLL1FM[INCSTP] = round[((2¹⁵ - 1) × 2 × 20) / (100 × 5 × 100)] = round[26.2136] = 26
- PLL1FM[MODPRD] × PLL1FM[INCSTP] = 100 × 26 = 2600 (which is less than 2¹⁵)
- MD (quantized) = ((100 × 26 × 100 × 5) / ((2¹⁵ - 1) × 20)) = 1.983703%

In this example, the modulation error is 0.0163%.

The recommended sequence for programming the FM is:

1. Power down PLL1 by writing PLL1CR[CLKCFG] = 00b.
2. Program PLL1FM while PLL1 is powered down.
3. Power up PLL1 by writing PLL1CR[CLKCFG] = 11b.

21.9 Maximum lock time

Maximum lock time is for the condition when the operating mode of the PLL has changed from power-down to Normal mode, input control bits are stable, and supplies (analog and digital) have attained levels shown in the *Data Sheet*. It is also the time in which the PLLs regain lock after a loss of lock event, assuming all inputs are stable and supplies are within specifications.

21.10 Initialization information

Coming out of reset PLL0 and PLL1 are disabled. The Dual PLLDIG initialization procedure is described in [PLLDIG initialization information](#) in the Clocking details.

Chapter 22

Clock Monitor Unit (CMU)

22.1 Chip-specific CMU information

The chip has nine instances of the CMU module. Each CMU module is programmed independently.

For more chip-specific information, see [Clock monitoring](#).

22.2 Introduction

The Clock Monitor Unit (CMU), also referred to as Clock Quality Checker or Clock Fault Detector, serves three purposes:

- Measures the frequency of clock source CLKMT0_RMN with CLKMN0_RMT as the reference clock
- Monitors CLKMN0_RMT frequency with CLKMT0_RMN as reference clock
- Monitors CLKMN1 frequency with CLKMT0_RMN as reference clock and detects if the monitored clock frequency leaves an upper or lower frequency boundary

NOTE

See the "Clocking" chapter for chip-specific sources used by the CMU.

One of the tasks is to supervise the integrity of the various clock sources on the chip, for example CLKMN0_RMT or CLKMN1. If the monitored clock frequency is less than the reference clock, or it violates an upper or lower frequency boundary, the CMU detects and reports this event. These events signal the FCCU, which can take the necessary corrective actions as its configuration dictates.

The CMU can monitor CLKMN0_RMT, which must have a frequency higher than that of CLKMT0_RMN divided by the factor shown in CMU_CSR[RCDIV], and reports this event. The CMU can also monitor CLKMN1 and generate an event if CLKMN1 is greater than a high frequency boundary or less than a low frequency boundary. The upper and lower frequency boundaries are defined by the CMU High Frequency Reference Register (CMU_HFREFR) and CMU Low Frequency Reference Register (CMU_LFREFR).

The second task of the CMU is to provide a frequency meter, which allows measuring the frequency of a clock source against a reference clock. This is useful to correct/calculate the time deviation of a counter that is clocked by the metered clocks.

NOTE

See the "Clocking" chapter for the number of CMU instances on this chip.

22.2.1 Main features

- CLKMT0_RMN frequency measurement with CLKMN0_RMT as reference clock.
- CLKMN0_RMT monitoring with respect to $\text{CLKMT0\_RMN} \div 2^{\text{CSR[RCDIV]}}$ clock.
- Upper or lower frequency boundary monitoring of CLKMN1 with respect to $\text{CLKMT0\_RMN} \div 4$.
- Event generation for various failures detected inside monitoring unit.

22.3 Block diagram

The block diagram of the CMU module(s) is shown in the "Clocking" chapter of this Reference Manual.

22.4 Signals

The table below describes the signals on the boundary of the CMU (in alphabetical order).

Table 22-1. Signal description

Signal	I/O	Description
CLKMNO_RMT	I	Monitored Clock Signal 0/Metered Clock Signal Reference — Receives a clock signal that the CMU compares to a specified low-limit frequency to determine whether the frequency of the clock signal is greater than the specified limit. Also provides a reference clock signal for all metered clock signals.
CLKMN1	I	Monitored Clock Signal 1 — Receives a clock signal that the CMU compares to specified low-limit and high-limit frequencies to determine whether the frequency of the clock signal is between the specified limits.
CLKMT0_RMN	I	Metered Clock Signal 0/Monitored Clock Signal Reference — Receives a clock signal that the CMU measures against a reference clock frequency. Also provides a reference clock signal for all monitored clock signals.

NOTE

See the "Clocking" chapter for device specific clock sources of each CMU.

22.5 Register description and memory map

This section describes in address order all the CMU registers. Each description includes a standard register diagram with an associated figure number. The CMU memory map is listed in the following table.

NOTE

See "Clocking" chapter for register and field availability details.

CMU memory map

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
0	CMU Control Status Register (CMU_CSR)	32	R/W	See section	22.5.1/850
4	CMU Frequency Display Register (CMU_FDR)	32	R	0000_0000h	22.5.2/851
8	CMU High Frequency Reference Register CLKMN1 (CMU_HFREFR)	32	R/W	0000_0FFFh	22.5.3/852
C	CMU Low Frequency Reference Register CLKMN1 (CMU_LFREFR)	32	R/W	0000_0000h	22.5.4/852
10	CMU Interrupt Status Register (CMU_ISR)	32	w1c	See section	22.5.5/853
18	CMU Measurement Duration Register (CMU_MDR)	32	R/W	0000_0000h	22.5.6/854

22.5.1 CMU Control Status Register (CMU_CSR)

Address: 0h base + 0h offset = 0h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	
R	0								SFM	0							
W																	
Reset	0	0	0	0	0	0	0	0	0*	0	0	0	0	0	0	0	

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	
R	0						CKSEL1		0						RCDIV		CME
W																	
Reset	0	0	0	0	0	0	0*	0*	0	0	0	0	0	0*	0*	0	

* Notes:

- RCDIV field: Not all CMU blocks will utilize this feature. See the “Clocking” chapter for CMU implementation details.
- CKSEL1 field: Not all CMU blocks will utilize this feature. See the “Clocking” chapter for CMU implementation details.
- SFM field: Not all CMU blocks will utilize this feature. See the “Clocking” chapter for CMU implementation details.

CMU_CSR field descriptions

Field	Description
0–7 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
8 SFM	Start frequency measure. The software can only set this bit to start a clock frequency measure. It is reset by hardware when the measure is ready in the CMU_FDR. NOTE: MSR[MD] must be written before enabling frequency measurement (CSR[SFM] = 1). Do not write MDR[MD] while CSR[SFM] = 1. 0 Frequency measurement is completed or not yet started 1 Frequency measurement is not completed
9–21 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
22–23 CKSEL1	Frequency measure clock selection bit. CKSEL1 selects the clock to be measured by the frequency meter. This only affects CMU instances that utilizes clock metering. Not all CMU blocks will utilize this feature. See the “Clocking” chapter for device specific CMU implementation details. 00 CLKMT0_RMN is selected 01 Reserved 10 Reserved 11 CLKMT0_RMN is selected
24–28 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

Table continues on the next page...

CMU_CSR field descriptions (continued)

Field	Description
29–30 RCDIV	CLKMT0_RMN division factor. These bits specify the CLKMT0_RMN division factor. The output clock frequency is $f_{\text{CLKMT0\_RMN}} \div 2^{\text{CMU\_CSR}[\text{RCDIV}]}$. This output clock is used as reference clock to compare with CLKMN0_RMT for crystal clock monitor feature. 00 CLKMT0_RMN $\div$ 1 (No division) 01 CLKMT0_RMN $\div$ 2 10 CLKMT0_RMN $\div$ 4 11 CLKMT0_RMN $\div$ 8
31 CME	CLKMN1 monitor enable. 0 CLKMN1 monitor is disabled 1 CLKMN1 monitor is enabled

22.5.2 CMU Frequency Display Register (CMU_FDR)

The CMU_FDR is used to determine the measured frequency of:

- CLKMT0_RMN

with respect to the reference clock CLKMN0_RMT.

NOTE

The CMU_FDR should be read only when
CMU_CSR[SFM] = 0.

Address: 0h base + 4h offset = 4h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0												FD																			
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	

CMU_FDR field descriptions

Field	Description
0–11 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
12–31 FD	Measured frequency bits. This register displays the measured frequency (f_{sel}) with respect to the reference clock ($f_{\text{CLKMN0\_RMT}}$). The measured value is given by the following formula: $f_{\text{sel}} = (f_{\text{CLKMN0\_RMT}} \times \text{CMU\_MDR}[\text{MD}]) \div \text{CMU\_FDR}[\text{FD}]$

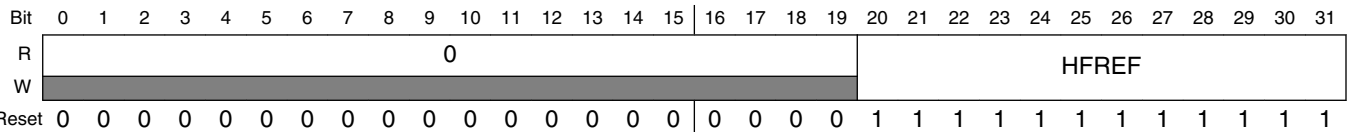
22.5.3 CMU High Frequency Reference Register CLKMN1 (CMU_HFREFR)

The HFREFR is configured for the high frequency reference that the CMU will use for comparing against the monitored clock. The figure and table below show the CMU_HFREFR register.

NOTE

For correct synchronization of CMU_HFREFR write data, the user must ensure that a time interval of 10 peripheral bus clock cycles + 40 CLKMT0_RMN clock cycles elapses before attempting subsequent writes to the same register.

Address: 0h base + 8h offset = 8h



CMU_HFREFR field descriptions

Field	Description
0–19 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
20–31 HFREF	High Frequency reference value. These bits determine the high reference value for the CLKMN1 frequency. The reference value is given by: $(\text{HFREF} \div 16) \times (f_{\text{CLKMT0\_RMN}} \div 4)$.

22.5.4 CMU Low Frequency Reference Register CLKMN1 (CMU_LFREFR)

The LFREFR is configured for the low frequency reference that the CMU will use for comparing against the monitored clock. The figure and table below show the CMU_LFREFR register.

NOTE

For correct synchronization of CMU_LFREFR write data, the user must ensure that a time interval of 10 peripheral bus clock cycles + 40 CLKMT0_RMN clock cycles elapses before attempting subsequent writes to the same register.

Address: 0h base + Ch offset = Ch

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0																LFREF															
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

CMU_LFREFR field descriptions

Field	Description
0–19 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
20–31 LFREF	Low Frequency reference value. These bits determine the low reference value for the CLKMN1 frequency. The reference value is given by: $(\text{LFREF} \div 16) \times (f_{\text{CLKMT0\_RMN}} \div 4)$.

22.5.5 CMU Interrupt Status Register (CMU_ISR)**NOTE**

All flags in the CMU_ISR are set asynchronously. This register must be read only after a CMU event is triggered. Otherwise, a read access on this register may fetch an incorrect value (see "Clocking" chapter for interrupt operation) .

NOTE

Before entering low-power stop modes, all clock frequency measurements in the CMU should be disabled. Failing to do so may result in ISR flags being spuriously set.

Address: 0h base + 10h offset = 10h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	Reserved															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	Reserved												FLCI	FHHI	FLLI	OLRI
W													w1c	w1c	w1c	w1c
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0*

* Notes:

- OLRI field: Not all CMU blocks will utilize this feature. See the "Clocking" chapter for specific CMU implementation details.

CMU_ISR field descriptions

Field	Description
0–27 Reserved	This field is reserved.
28 FLCI	CLKMN1 frequency less than reference clock event status. This bit is set by hardware when CLKMN1 frequency becomes lower than reference clock frequency ($f_{\text{CLKMTO\_RMN}} \div 4$) value and CLKMN1 is 'ON'. It can be cleared by software by writing '1'. NOTE: This bit will show indeterministic behavior if the clock monitored by the CMU is not enabled . 0 No FLC event 1 FLC event occurred
29 FHHI	CLKMN1 frequency higher than high reference event status. This bit is set by hardware when CLKMN1 frequency becomes higher than the high frequency reference value determined by CMU_HFREFR[HFREF] and CLKMN1 is 'ON'. It can be cleared by software by writing '1'. 0 No FHH event 1 FHH event occurred
30 FLLI	CLKMN1 frequency less than low reference event status. This bit is set by hardware when CLKMN1 frequency becomes lower than the low frequency reference value determined by CMU_LFREFR[LFREF], and CLKMN1 is 'ON'. Software clears this field by writing a '1'. NOTE: This bit will show indeterministic behavior if the clock monitored by the CMU is not enabled . 0 No FLL event 1 FLL event occurred
31 OLRI	Oscillator frequency less than $f_{\text{CLKMTO\_RMN}} \div 2^{\text{CMU\_CSR[RCDIV]}}$ event status. This bit is set by hardware when the $f_{\text{CLKMN0\_RMT}}$ is less than $f_{\text{CLKMTO\_RMN}} \div 2^{\text{CMU\_CSR[RCDIV]}}$ frequency and CLKMN0_RMT is 'ON' . It can be cleared by software by writing '1'. NOTE: While entering STOP mode, OLRI may be triggered due to stoppage of XOSC. To avoid this, disable the XOSC before entering STOP mode. 0 No OLR event 1 OLR event occurred

22.5.6 CMU Measurement Duration Register (CMU_MDR)

Address: 0h base + 18h offset = 18h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0												MD																			
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	

CMU_MDR field descriptions

Field	Description
0–11 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
12–31 MD	Measurement duration bits This field displays the measurement duration in terms of selected clock (CLKMT0_RMN) cycles. This value is loaded in the frequency meter down-counter. The down-counter starts counting when CMU_CSR[SFM] = 1. NOTE: Only values that do not cause a CMU_FDR[FD] overflow shall be written to CMU_MDR[MD]. There is no internal status of a CMU_FDR[FD] overflow.

22.6 Functional description

22.6.1 Frequency meter

The purpose of the frequency meter is to evaluate the deviation from the nominal metered source (such as CLKMT0_RMN) frequencies. Programming of the CMU_CSR[CKSEL1] field is used to select one of the metered clocks from a multiplexer that drives a simple Frequency Meter (see the CMU Block Diagram in the "Clocking" chapter). The reference clock for the Frequency Meter is the CLKMN0_RMT signal. The measurement starts when CMU_CSR[SFM] = 1. The measurement duration is given by the contents of CMU_MDR[MD] in terms of number of clock cycles of the selected metered clock. The CMU_CSR[SFM] bit is cleared by hardware once the frequency measurement is complete and the count is loaded in CMU_FDR[FD]. The frequency of the selected clock (f_{sel}) can be derived from the value loaded in the CMU_FDR[FD] as shown in the following equation:

$$f_{sel} = f_{CLKMN0\_RMT} \times \frac{CMU\_MDR[MD]}{CMU\_FDR[FD]}$$

22.6.2 CLKMN0_RMT supervisor

If frequency of CLKMN0_RMT is smaller than the frequency of CLKMT0_RMN $\div 2^{CMU\_CSR[RCDIV]}$ and CLKMN0_RMT is 'ON', then:

- The CMU writes 1 to CMU_ISR[OLRI].
- The CMU asserts the OLR signal.

Note

$f_{\text{CLKMN0\_RMT}}$ must be greater than $f_{\text{CLKMT0\_RMN}} \div 2^{\text{CMU\_CSR[RCDIV]}}$ by at least 0.5 MHz in order to guarantee correct $f_{\text{CLKMN0\_RMT}}$ monitoring.

22.6.3 CLKMN1 supervisor

The frequency of CLKMN1 (f_{CLKMN1}) can be monitored by programming CMU_CSR[CME] = 1. CLKMN1 monitoring starts as soon as CMU_CSR[CME] = 1. This monitor can be disabled at any time by programming CMU_CSR[CME] = 0.

If f_{CLKMN1} is greater than the reference value determined by fields CMU_HFREFR[HFREF] and CLKMN1 is 'ON', then:

- The CMU writes 1 to CMU_ISR[FHHI].
- The CMU asserts the FHH signal.

If f_{CLKMN1} is less than a reference clock frequency ($f_{\text{CLKMT0\_RMN}} \div 4$) and the CLKMN1 is 'ON', then:

- The CMU writes 1 to CMU_ISR[FLCI].
- The CMU asserts the FLC signal.

If f_{CLKMN1} is less than a reference value determined by the bits CMU_LFREFR[LFREF] and the CLKMN1 is 'ON', then:

- The CMU writes 1 to CMU_ISR[FLLI].
- The CMU asserts the FLL signal.

Note

f_{CLKMN1} must be greater than $f_{\text{CLKMT0\_RMN}} \div 4$ by at least 0.5 MHz in order to guarantee correct CLKMN1 monitoring.

An example of determining the $\text{HFREF}_{\text{Actual}}$ is as follows. Assume a $f_{\text{CLKMT0\_RMN}} = 16$ MHz with a accuracy of $\pm 5\%$ (see *Data Sheet* for actual value). In order to monitor a $f_{\text{CLKMN1}} = 200$ MHz, the ideal $\text{HFREF}_{\text{Ideal}} = 800$. The actual HFREF value will be 842 when the accuracy is taken into

consideration ($\text{HFREF}_{\text{Actual}} = (\text{HFREF}_{\text{Ideal}} \div 0.95)$). The actual
LFREF value will be 762 when accuracy is taken into
consideration ($\text{LFREF}_{\text{Actual}} = \text{LFREF}_{\text{Ideal}} \div 1.05$).

Chapter 23

Embedded Memories

23.1 Overview

This chip's features of embedded memory include:

- On-chip SRAM, including system SRAM and local instruction and data caches for each processor core
- On-chip system error correction code (ECC) flash memory
- End-to-end ECC (e2eECC) error detection and correction (not all modules support e2eECC)
- Embedded memories in select peripherals, including:
 - FlexCAN
 - M_CAN
 - eDMA
 - FEC
 - eTPU
 - CSE

For more information about embedded memory in a peripheral, see the peripheral's dedicated chapter.

23.2 End-to-end Error Correction Code (e2eECC)

To support market requirements related to improved functional and transient fault detection capabilities, this chip includes end-to-end ECC (e2eECC) support. This e2eECC is structurally different from traditional "ECC at memory" functionality because

it provides robust error detection capabilities from one endpoint of an information transfer to another endpoint, with temporary information storage in one or more intermediate components.

Memory protected by ECC/EDC traditionally generates and checks additional error parity information local to the memory unit to detect and/or correct errors that have occurred on data stored in the memory. In contrast, e2eECC generates error protection codes at the source of data generation. It sends the encoded data and error protection codes to intermediate storage when a memory write is initiated by a bus master, and it performs a data integrity check using the previously stored error protection codes at a data memory when a read of stored information is requested by a bus master. Depending on the particular memory unit, the intermediate storage may transform the generated error protection codes into another format for storage and then regenerate the codes for provision when a request is made to read the stored information, or it may simply store the original protection codes unaltered.

To protect additional information associated with an access, the error protection codes are generated on the basis of more than the data associated with a storage location. In particular, address information corresponding to the access location of the stored information is combined with the stored data at the data source to generate error protection codes that cover certain types of addressing errors in the system interconnection or in the memory unit. The error protection codes may be checked at the memory unit on a store operation to ensure that no address information or data has been corrupted while the request has transitioned through the device from the bus master source. The memory unit may simply store the received data and protection codes at the address it receives, or it may check the error protection codes to ensure that address decoding within the storage memory was performed properly (although not all address decoding errors can be detected this way).

On a read request from a bus master, the memory unit retrieves the data information and error protection codes corresponding to the received address from the storage location or locations and supplies the data along with the error protection codes to the requesting device. The requesting device uses a locally stored address value corresponding to the read request to check the returning data and error protection codes to ensure that no errors have occurred in either addressing the memory or in the retrieved data. This approach ensures that the address sent for the request was not corrupted, the addressed location was actually accessed (to the extent it is possible to ensure), and the stored data was error free, to the extent the ECC coding scheme is able to detect. As a result, the fault coverage provided by the end-to-end check is considerably more robust than the previous implementation of locally generated and checked/corrected error protection at each memory unit.

The following table compares the traditional and e2eECC approaches.

Table 23-1. ECC comparison of data write-then-read sequence

Traditional ECC	End-to-end ECC (e2eECC)
Bus master initiates data write	Bus master initiates data write and generates ECC checkbits based on 29-bit address and 64-bit data fields for the computational shell, or based on 30-bit address and 32-bit data field for the peripheral shell
Data write transfer routed from bus master to appropriate bus slave	Data write transfer (including checkbits) routed from bus master to appropriate bus slave
For a bus memory slave, generate the ECC checkbits based on the data value and store data + checkbits into the memory	For a bus memory slave, store data and checkbits into the memory
Bus master initiates data read of previously written memory location	Bus master initiates data read of previously written memory location
Data read transfer routed from bus master to appropriate bus slave	Data read transfer routed from bus master to appropriate bus slave
For a bus memory slave, the memory array is accessed, the controller performs the ECC checkbit decode and syndrome generation, performs any needed single-bit correction, and drives the read data onto the system bus interconnect	For a bus memory slave, the memory array is accessed, and the controller passes the read data and associated checkbits onto the system bus interconnection
The bus master captures the read data and continues	The bus master captures the read data and associated checkbits, performs the ECC checkbit decode and syndrome generation, performs any needed single-bit correction, and continues

The scope of differences in the operations "covered" by the ECC checks is readily apparent with the e2eECC concept, improving fault detection capabilities in two important aspects:

- The entire data transaction—from the initiating bus master, through the entire platform crossbar steering mechanism, to the destination slave target—is covered during write accesses. Likewise, a read is checked from the initiating bus master, through the crossbar steering mechanism, through the actual memory read and transmission of the data and checkbits back to the bus master, where the integrity and correctness of the entire transaction is checked.
- The selected ECC protects both the address field and the data field.

NOTE

Compared to MPC5676R: For every SRAM and DMA transfer initiation, e2eECC on MPC5777C requires 2 additional clock cycles. The performance impact of these additional clocks can be reduced through the use of bursting.

In general, e2eECC improves implementation. The traditional ECC approach places the checkbit decode and error syndrome generation with the error/no_error state, affecting whether the system bus transfer must be stalled (to correct a single bit error or report a

noncorrectable event) or is allowed to complete (for error-free transfers) in a critical timing arc that often sets the upper limit of the operating frequency of the microcontroller. With the e2eECC scheme, the checkbit decode and error syndrome logic is located in the requesting bus master with generally more degrees of implementation freedom; the error/no_error state determination is removed from the system bus cycle termination logic, producing an improved timing arc and generally higher operating speeds.

Errors that occur within the system interconnect are typically manifested as an incorrect address, incorrect write data, or incorrect read data to be presented to the slave or back to the master. Errors occurring within the storage typically manifest themselves eventually in corrupted read data or checkbits being returned to a requester.

The following diagram provides an overview of the e2eECC implementation on this chip.

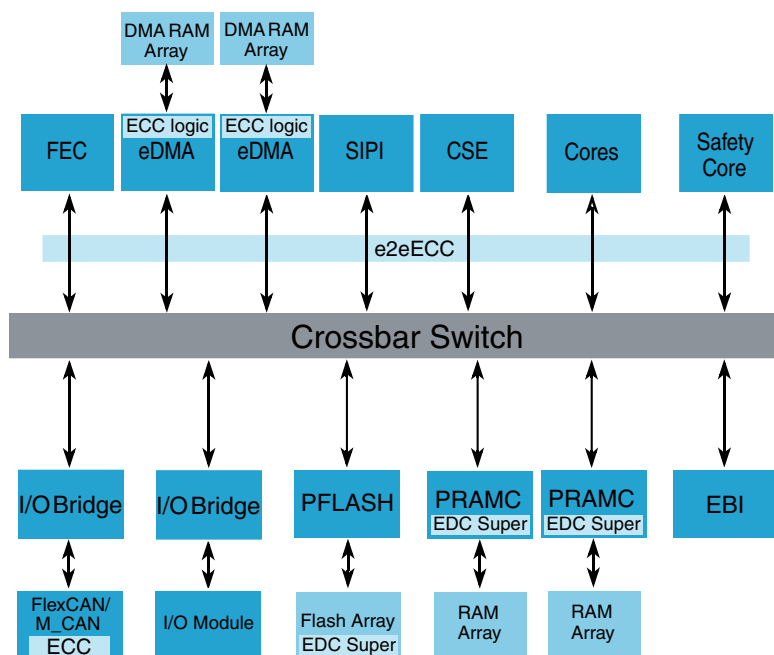


Figure 23-1. e2eECC overview

NOTE

Due to the e2eECC implementation as it relates to the cores, treat ECC errors discovered during cache line fills as imprecise.

Chapter 24

RAM Controller (PRAMC)

24.1 Chip-specific PRAM information

This chip has two instances of the PRAM module.

24.1.1 PRAM optional read wait states

The following table shows programming values for the BO_DIS and FT_DIS fields of the PRAM's PRCR1 register and the resulting access timing for a WRAP4 burst. For additional information, see [Optional read wait-state](#).

Table 24-1. Read wait states for WRAP4 burst

BO_DIS	FT_DIS	WRAP4 timing
0	0	1-1-1-1
0	1	2-1-1-1
1	0	1-1-1-1
1	1	2-2-2-2

24.1.2 Reset effects on System RAM accesses

If any of the following types of reset event occurs during a read or write operation to System RAM, the completion of that access depends on the cycle at which the reset occurs. Data read from or written to System RAM before the reset event occurred is retained, but no other address locations are accessed or changed.

- External reset via RESET pin
- FCCU reset
- JTAG reset

- Software system reset
- Dual core reset

If the System RAM is cached, cache lines might retain indeterminate data that is not written to memory unless the region is set for writethrough mode.

NOTE

Standby memory might contain the previous data values if a reset occurs while the cache is running in copyback mode.

For descriptions of reset types, see [Reset sources](#).

24.1.2.1 Other reset types

The software external reset does not affect internal logic and therefore does not cause RAM corruption for any RAMs.

For all other reset types, no System RAM retention is guaranteed.

24.1.2.2 Other RAMs

For all other RAMs on the chip, no retention is guaranteed for any reset type.

24.2 Introduction

This section provides an overview of the Platform RAM Controller. The RAM controller acts as an interface between the system bus (AHB-Lite 2.v6) and the integrated system RAM. It converts the protocols between the system bus and the dedicated RAM array interface.

The RAM controller supports two 64-bit AHB interfaces and a 64-bit RAM array interface. The primary AHB port provides a connection to the platform crossbar for direct RAM access from the various crossbar masters. Shown below in [Figure 24-1](#) is a simplified block diagram depicting the position of the RAM controller within a typical platform architecture.

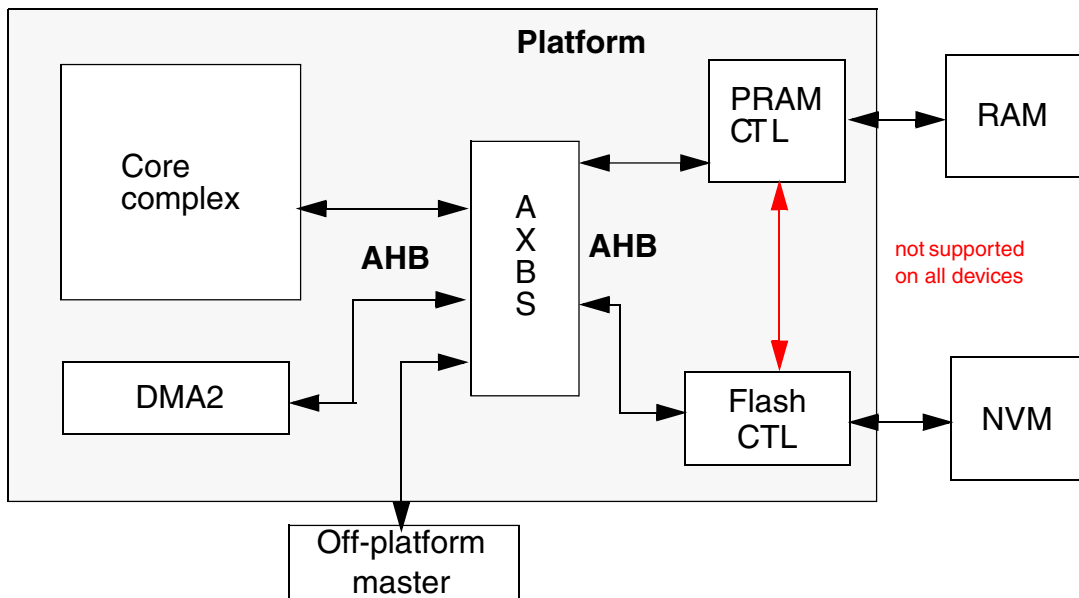


Figure 24-1. Simplified platform block diagram

The following list summarizes the key features of the RAM controller:

- System bus supports 64-bit data + 8-bit ECC AHB interfaces
- Array interface supports a 64-bit data + 8-bit ECC interface
- Late-write support via 64-bit data + 8-bit ECC storage buffer to support single-cycle write accesses
- Configurable read access timing (zero or one wait-state programmable) allowing use in large range of frequency targets
- Read-modify-write operation to support array write size less than a doubleword

The address path of the RAM controller contains a mux that chooses among the addresses presented on the system bus for a read request, either the address from the temporary holding register or the address from the late-write buffer. The temporary holding register contains the address which was presented during the AHB address phase of the write request. The request is stalled from being presented to the RAM by one cycle in order to present simultaneously the address and associated write data, which is not valid until the AHB dataphase. The late-write buffer holds write requests which are delayed to facilitate single-cycle response on the system bus.

The read datapath contains a mux that chooses the source of the read data to be presented on the system bus on a read request. In the event that a read request matches the contents of the late-write buffer, a RAM access is not required and the late-write buffer contents are selected onto the read databus. Otherwise, the read data is a result of a RAM access. Along with the read and write data, the corresponding ECC codewords traverse the entire datapath of the RAM controller, including the late-write storage buffer, to support end-to-end ECC (e2eECC) coverage.

The write datapath contains the mux that chooses the source of the write data as well as the control logic for generating read-modify-write operations on writes that are less than 64 bits in size. A write operation can be performed to empty the contents of the late-write buffer. In the case of back-to-back writes, the write may be forced to bypass the late-write buffer and instead be stored directly, precisely in the RAM.

24.3 SRAM controller memory map and register definition

The RAM controller module provides an IPS programming model mapped to a standard on-platform peripheral slot. The programming model can only be referenced using a 32-bit (word) access. Attempted references using different access sizes or to undefined (reserved) addresses generate an IPS error termination.

NOTE

Attempted updates to the PRAMC programming model while a PRAMC operation is in progress will result in non-deterministic behavior. Software must be architected to avoid this scenario by ensuring that PRAMC configuration changes are made only during system boot or when only one master is enabled. In multi-core devices, update the PRAMC configuration only when one core is active and no other masters, e.g., DMA or communications modules, are enabled. Move any instruction execution or memory references outside the system RAM while updating the PRAMC configuration, e.g., to the core local memory space.

PRAMC memory map

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/page
0	Platform RAM Configuration Register 1 (PRAMC_PRCR1)	32	R/W	0000_0100h	24.3.1/866

24.3.1 Platform RAM Configuration Register 1 (PRAMC_PRCR1)

The Platform RAM Configuration Register 1 (PRCR1) is used to specify operation of the RAM controller.

NOTE

This register is accessible only in supervisor mode. Accesses in user mode return a transfer error.

Address: 0h base + 0h offset = 0h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31		
R	0						Reserved		Reserved	P0_ BO_ DIS	0						FT_ DIS	
W							Reserved		Reserved	P0_ BO_ DIS							FT_ DIS	
Reset	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0		

PRAMC_PRCR1 field descriptions

Field	Description
0–21 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
22–23 Reserved	This field is reserved.
24 Reserved	This field is reserved.
25 P0_BO_DIS	Port p0 read burst optimization disable. NOTE: The number of cycles taken for a RAM access can vary ± 1 clock cycle depending on the RAM speed relative to the PRAMC controller clock frequency. If system RAM is running at the same frequency as the PRAMC controller, a random initial access takes 2 clock cycles. If system RAM is running at a slower frequency, a random initial access may take 3 clock cycles. Subsequent burst beats take either 1 or 2 cycles depending on RAM speed relative to the PRAMC controller clock frequency. 0 64-bit WRP4 read bursts are optimized such that the controller returns a 2-1-1-1 response when PRCR1[FT_DIS]=1 1 64-bit WRP4 read bursts are not optimized; the controller returns a 2-2-2-2 response when PRCR1[FT_DIS]=1
26–30 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
31 FT_DIS	Flow through disabled. This field defines the AHB response on reads. The state of this field has no impact on the response latency on writes. This bit is cleared by hardware reset. NOTE: Do not change the FT_DIS bit value while accessing system RAM. Relocate code programming the FT_DIS bit to another memory area, e.g., local core memory. 0 RAM read data is passed directly to the system bus, incurring no additional latency 1 RAM read data is registered prior to returning on the system bus, incurring 1 extra cycle of latency

24.4 Functional description

This section describes the functions of the RAM controller.

24.4.1 Read and Write operations

The RAM controller processes read and write requests to on-chip RAM and provides a register interface for access to performance tuning functions.

NOTE

The RAM controller register interface is accessible only in supervisor mode. Accesses in user mode return a transfer error.

24.4.1.1 Reads

Read transfers, of any size, can be configured to complete with a zero or one additional wait state response on the system bus.

24.4.1.2 Optional read wait-state

The RAM controller can be optionally programmed to register RAM read data prior to returning it on the system bus. Setting the `PRCRx[FT_DIS]` field inserts a register in the read data path for use when operating the controller at high frequencies. The state of `PRCRx[FT_DIS]` field has no effect on writes.

A random, initial access will take 2 clock cycles (2T) to complete if `PRCRx[FT_DIS]=0`, and a WRAP4 burst will have an access time of 2-2-2-2. A random, initial access will take 3 clock cycles (3T) to complete if `PRCRx[FT_DIS]=1`, and a WRAP4 burst will have an access time of 3-2-2-2.

NOTE

The number of cycles taken for a RAM access can vary ± 1 clock cycle depending on the RAM speed relative to the PRAMC controller clock frequency. If system RAM is running at the same frequency as the PRAMC controller, a random initial access takes 2 clock cycles. If system RAM is running at a slower frequency, a random initial access may take 3 clock cycles. Subsequent burst beats take either 1 or 2 cycles

depending on RAM speed relative to the PRAMC controller clock frequency.

24.4.2 Writes

This section discusses various write operations of the RAM controller.

24.4.2.1 64-bit writes

Aligned 64-bit writes execute in a single AHB data phase cycle, allowing for zero wait states on back-to-back writes of these sizes. If, during the data phase of a write, a read is requested on the AHB, the write is registered in the late-write buffer, allowing the read to take place without a wait state. The valid buffered or late-write data is stored on the next available array address phase.

Back to back 64-bit writes execute slightly differently whereby the first write bypasses the late-write buffer. Rather, the write data is stored directly to the array in the same cycle in which it is valid on the AHB.

24.4.2.2 Less than 64-bit writes

Writes of size less than 64 bits incur a read-modify-write action as a consequence of the ECC coding scheme's 64-bit granularity. In the case of a read-modify-write action, the RAM controller performs SEC/DED on the read data. The write data is merged into the appropriate byte lanes along with the potentially corrected read data. A new codeword is generated based on the newly formed doubleword. The newly formed doubleword and its associated checkbits are subsequently written to the RAM. [Figure 24-3](#) provides details on the read-modify-write datapath. For details on the ECC coding scheme, refer to [Reliability considerations](#).

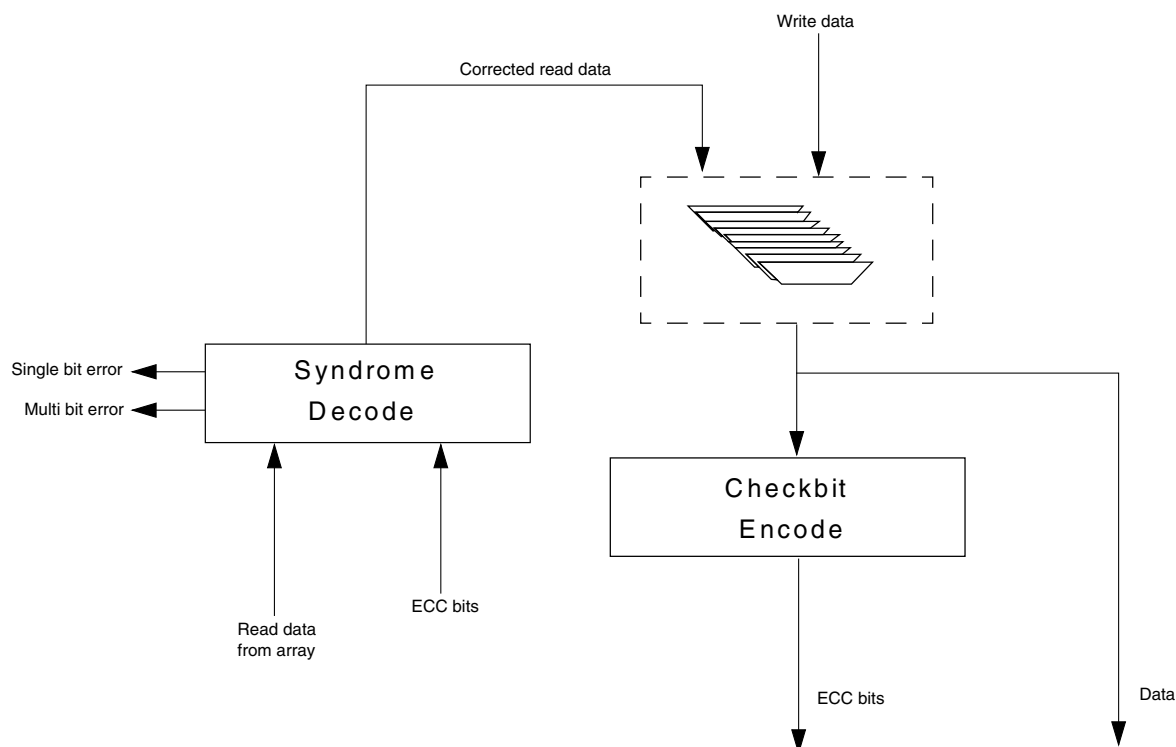


Figure 24-3. Read-modify-write datapath

On a read-modify-write operation, the array read data is registered after it is decoded and before it is merged with the AHB write data. Therefore, writes of size less than 64 bits require the insertion of one wait state before the data phase can be completed.

24.4.2.3 Unaligned writes

The RAM controller is compliant with the AMBA-AHB2.v6 Extensions specification with regard to byte strobes. The size of the transfer is sufficient to cover all bytes being written and covers more bytes in the case of an unaligned transfer. The address of the transfer is rounded down to the nearest boundary of the size of the transaction.

NOTE

Unaligned writes always generate read-modify-write operations in the RAM controller in order to preserve the validity of the ECC codeword.

24.5 Initialization/application information

It is essential that each memory address be written to a known value before it is read, to initialize the ECC. This includes reads generated from the read-modify-write operation which occurs when a write transfer of less than 64 bits or an unaligned write is requested. Without writing an address to a known value first, a read from this address will most likely generate an uncorrectable ECC event.

One possibility for initializing PRAMC memory space is to use a stored 64-bit word instruction such as Store Multiple Word (`e_stmw`) in Power Architecture VLE.

24.6 Reliability considerations

This section discusses reliability considerations of the RAM controller.

24.6.1 Hsiao ECC algorithm

The e2eECC scheme implements a single-error correction, double-error detection (SECDED) code using the so-called Hsiao odd-weight column criteria. These codes are named for M.Y. Hsiao, an IBM researcher who published extensively in the early 1970s on SECDED codes better suited for implementation in protecting (mainframe) computer memories than traditional Hamming codes.

The Hsiao codes are Hamming distance 4 implementations which provide the SECDED capabilities. The minimum odd-weight constraints defined by Hsiao are relatively simple in the resulting implementation of the parity check H matrix which defines the association between the data (and address) bits and the checkbits. They are:

1. There are no all-zeroes columns.
2. Every column is distinct.
3. Every column contains an odd number of ones, and hence is "odd weight".

In defining the H-matrix for this family of devices, these requirements from Hsiao were applied. Additionally, there are a variety of ECC codeword requirements associated with specific functional requirements associated with the system RAM that further dictated the specific column definitions. In any case, the resulting ECC is organized based on 64 data bits plus 29 address bits (the upper bits of the 32-bit address field minus the 3 bits which select the byte within the 64-bit (8-byte) data field).

Table 24-4. RAM controller basic H-matrix definition

[illegible]

1. Bit numbering is AHB convention: bit 0 is LSB. D[7:0] corresponds to byte at address 0. D[63:56] corresponds to byte at address 7.

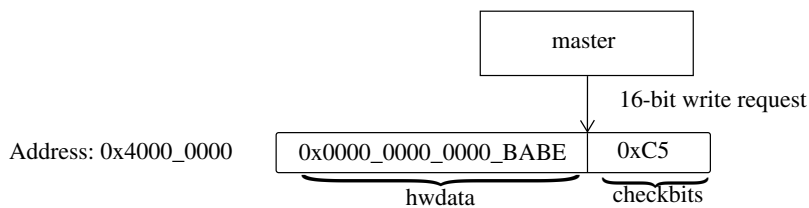
2. Bit numbering is AHB convention: bit 0 is LSB. D[7:0] corresponds to byte at address 0. D[63:56] corresponds to byte at address 7.

24.6.1.1 e2eECC considerations on less-than-64-bit write transactions

Recall writes of size less than 64-bit incur a read-modify-write action as a consequence of the ECC coding scheme 64-bit granularity. In the case of a read-modify-write action, the RAM controller performs SEC/DED on the read data. The RAM controller uses the following checkbit manipulation technique to ensure any faults in the PRAMC control logic associated with performing the read-modify-write action will ultimately be detectable by e2eECC:

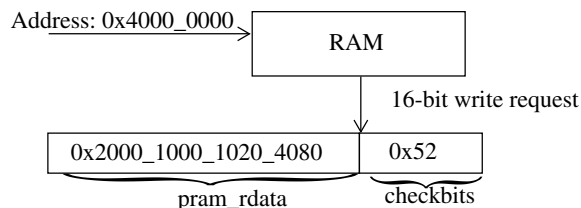
- On an 8-, 16-, or 32-bit write request, the write data is presented by the master on the AHB bus in the correct byte lanes, with the non-pertinent byte lanes zeroed out, and the associated checkbit is based on this "zero-padded" write data.

Consider the following example whereby the master presents a 16-bit write request at address 0x40000000:

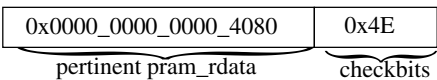


- The RAM controller detects that the request is a less-than-64-bit write request and initiates a read to address 0x40000000. An ECC check and potential single-bit correction is performed on the data returned from the RAM.

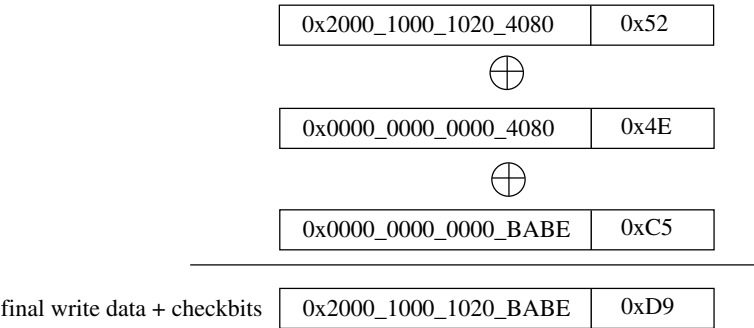
If a non-correctable ECC event is detected, the AHB request is terminated with error on the system bus.



- The RAM controller isolates the pertinent byte lanes of the read data by zero-padding the non-pertinent byte lanes and calculating the checkbit contribution associated with the read data in the byte lanes to be updated as a result of the write request.



- Merge the non-pertinent read data bytes with the write data in the appropriate byte lanes. Take the checkbit value that was stored in the RAM and remove the checkbit contribution associated with the read data in the byte lanes to be updated. Then introduce the checkbits associated with the write data. The result is ultimately a checkbit value associated with the complete 64-bit data to be written to the array.



This technique calculates the ultimate checkbit value on a read-modify-write transaction through a series of data manipulations, taking care to avoid discarding the original checkbits provided by the requesting master, such that faults in the crossbar (XBAR) and RAM controller datapath and control logic are covered by e2eECC.

Malfunction of the ECC logic described above may result in the corruption of the SEC/DED event reporting. The RAM controller performs EDC after ECC check on all read-modify-write transactions. If a mismatch is detected, indicating a failure in the ECC logic, the event is reported to the FCCU module.

24.6.2 Transaction monitor

The integrity of the address and data on a system RAM transaction is covered by e2eECC check performed by the requesting master. Fault detection coverage of the address path and control within the RAM controller is handled by a transaction monitor which verifies the integrity of the transactions between the RAM controller and the RAM array. The transaction monitor verifies RAM transactions initiated to service the following types of transactions:

- Direct system bus read or write request
- Read followed by write to fulfill a read-modify write transaction
- Write transaction to empty the late-write buffer

The function of the transaction monitor relies on a feedback path between the RAM controller and RAM array, wherein the RAM provides latched address and control feedback outputs as an indication of received inputs when a RAM access is initiated.

This feedback information is used by the RAM controller transaction monitor to verify the expected transaction. If a mismatch is detected, indicating a failure in the address generation or control logic within the RAM controller or the transmission path between the RAM controller and RAM array, the event is forwarded to the Fault Collection and Control Unit (FCCU).

Since writes to the RAM can be buffered, the transaction monitor also tracks the contents of the late-write buffer. When the late-write buffer is emptied, outputs from the RAM are evaluated against the expected contents of the late-write buffer as tracked by the transaction monitor.

As a countermeasure against false misses to the late-write buffer on reads, the transaction monitor also verifies the expected source of data returned on a read as supplied either from the RAM or from the late-write buffer.

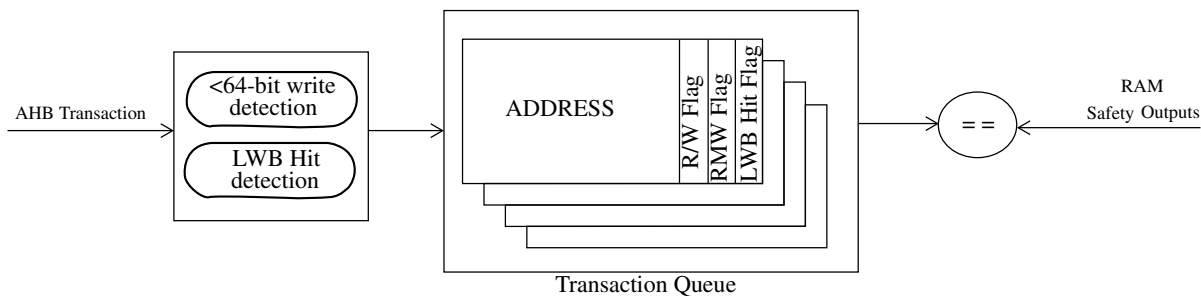


Figure 24-4. RAM controller transaction monitor block diagram

Chapter 25

Flash Memory Controller (PFLASH)

25.1 Chip-specific PFLASH information

The PFLASH registers PFCR1, PFCR2, and PFAPR control the interaction of master modules with the flash memory array by enabling/disabling prefetch or by controlling access to the array on a per-master basis.

Fields in these registers refer to the masters' logical ID numbers. For example, the PFAPR[M2AP] field controls flash memory array access by logical master 2, and the PFAPR[M3AP] field controls flash memory array access by logical master 3.

- For the list of bus masters' logical IDs, see [XBAR master and slave assignments](#).
- For logical master ID numbers that are unused on this chip, the corresponding PFLASH register fields are reserved.

For more information about configuring the PFLASH, see the [Platform Flash Configuration Register 3 \(PFLASH_PFCR3\)](#) details.

25.2 Introduction

The flash memory controller has these functions:

- It acts as an interface between the system bus (AHB-Lite 2.v6) and the flash memory array.

The flash memory controller supports two 64-bit AHB buses and a 256-bit read data interface from each flash memory array. Each AHB port contains a 4-entry, 4-way set-associative mini-cache as well as an associated controller that prefetches sequential lines of data from the flash arrays into the mini-cache. This buffer mechanism serves to deliver flash read data with zero-wait state response on lines that reside in the cache. AHB requests that miss the cache generate the needed flash array access and are forwarded to

the AHB upon completion. Each mini-cache entry is 256 bits in size, matching the flash array page size and providing 512 bytes of total buffered storage. Along with the read and write data, the corresponding ECC codewords traverse the entire datapath of the flash memory controller, including the mini-cache, to support end-to-end ECC (e2eECC) coverage.

25.3 Features

The following list summarizes the key features of the flash memory controller:

- Two 64-bit AHB interface ports (p0, p1) allowing simultaneous access to dedicated prefetch mini-cache per slave port.
- 256-bit read data bus + 64-bit write data bus
- Configurable read buffering and line prefetching support via 4-entry, 4-way set-associative mini-cache plus prefetch controller per AHB port to provide single-cycle "buffer hit" read response.
- Configurable access control based on read/write and AHB master ID attributes.
- Configurable access timing (wait-state programmable) allowing use in a wide range of frequency targets.
- Optional address pipelining capability to maximize throughput.
- Support for reporting of single- and multi-bit flash ECC events on a 64-bit doubleword boundary.

25.4 Block diagrams

[Figure 25-1](#) illustrates the connections of the flash memory controller and flash memory array within the chip.

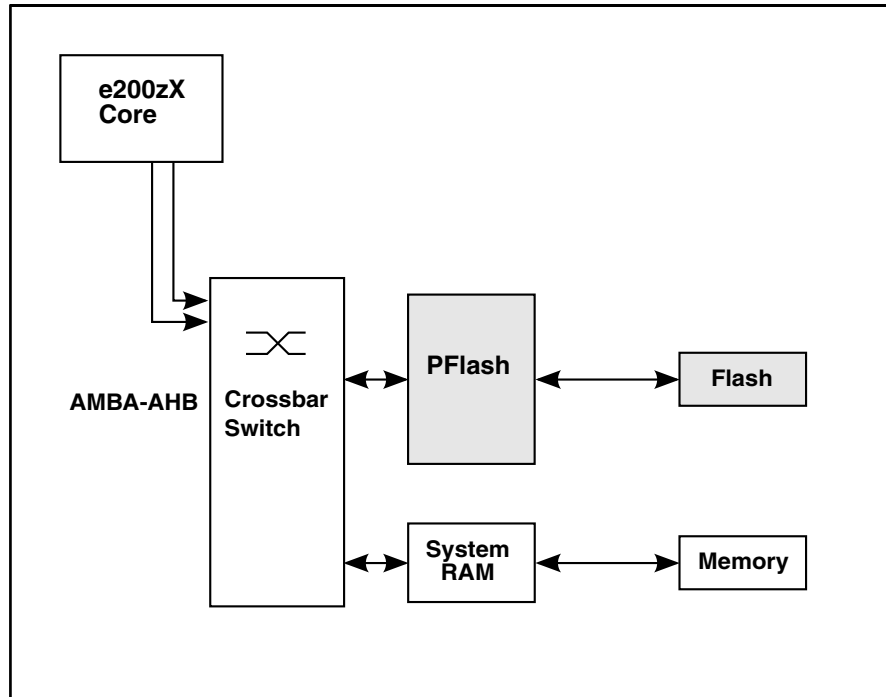


Figure 25-1. Platform-centric simple block diagram with flash memory controller

25.5 Flash memory controller memory map

The flash memory controller module provides an IPS programming model mapped to a standard 16 KB on-platform peripheral slot. The programming model consists of flash memory access configuration.

The programming model can only be referenced using a 32-bit (word) access. Attempted references using different access sizes or to undefined (reserved) addresses or in user mode generates an IPS error termination. The flash controller allows access to the programming model by all system bus masters.

The programming model can only be accessed in supervisor mode.

A write to any flash memory controller register, when executed concurrently with a flash memory access, will result in non-deterministic behavior and may include lockup of the flash memory controller. There is no idle indicator for the flash controller, so software must be architected to prevent any possible conflicts. The following recommendations will help.

- For multi-core devices, start only one core and execute initialization code to completion before starting the remaining cores.

- If flash must be reconfigured during application code execution, temporarily move all required code and data to system RAM before any writes to flash memory controller registers.
- If application software functionality includes invalidating a port read buffer, the recommended method for invalidating a port read buffer is:
 - a. Unlock the buffer's enable field by setting the PFLASH_PFCR3[BFEN_LK] field to '0'
 - b. Disable the buffer by setting it's associated enable field (PFLASH_PFCR1[Pn_BFEN]) to '0'
 - c. If needed, enable the buffer by setting it's associated enable field (PFLASH_PFCR1[Pn_BFEN]) to '1'

The above method requires updating flash memory controller registers, so software must ensure that no flash accesses are issued concurrently with any of the steps. In cases where that cannot be guaranteed, there is a workaround but its use is not generally recommended because it relies on a hardware side effect instead of the flash memory controller's built-in functions intended for invalidating port read buffers. The workaround is to issue an interlock write without an accompanying program or erase operation—an interlock write, without an accompanying program or erase, does not change any flash content.

NOTE

1. If this workaround is implemented in application code, note that an interlock write invalidates only the buffers associated with the system bus port on which the interlock write was issued.
2. See the flash memory chapter for the steps to issue an interlock write.
3. See the flash memory controller's chip-specific information for any restrictions on the interlock write data size.

PFLASH memory map

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/page
0	Platform Flash Configuration Register 1 (PFLASH_PFCR1)	32	R/W	0000_0601h	25.5.1/881
4	Platform Flash Configuration Register 2 (PFLASH_PFCR2)	32	R/W	0000_0001h	25.5.2/884
8	Platform Flash Configuration Register 3 (PFLASH_PFCR3)	32	R/W	0000_0000h	25.5.3/887
C	Platform Flash Access Protection Register (PFLASH_PFAPR)	32	R/W	FFFF_FFFFh	25.5.4/889

25.5.1 Platform Flash Configuration Register 1 (PFLASH_PFCR1)

The PFlash Configuration Register 1 (PFCR1) controls the operation of Port p0 of the PFLASH_C55FM flash memory controller.

NOTE

See the chip-specific platform flash controller information for information on the masters.

Address: 0h base + 0h offset = 0h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	P0_M15PFE	P0_M14PFE	P0_M13PFE	P0_M12PFE	P0_M11PFE	P0_M10PFE	P0_M9PFE	P0_M8PFE	P0_M7PFE	P0_M6PFE	P0_M5PFE	P0_M4PFE	P0_M3PFE	P0_M2PFE	P0_M1PFE	P0_M0PFE
W	P0_M15PFE	P0_M14PFE	P0_M13PFE	P0_M12PFE	P0_M11PFE	P0_M10PFE	P0_M9PFE	P0_M8PFE	P0_M7PFE	P0_M6PFE	P0_M5PFE	P0_M4PFE	P0_M3PFE	P0_M2PFE	P0_M1PFE	P0_M0PFE
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	APC			0	RWSC				0	P0_DPFE	0	P0_IPFE	0	P0_PFLIM		P0_BFEN
W																
Reset	0	0	0	0	0	1	1	0	0	0	0	0	0	0	0	1

PFLASH_PFCR1 field descriptions

Field	Description
0 P0_M15PFE	Port0 Master 15 Prefetch enable. This bit controls whether prefetching may be triggered by AHB master 15. 0 No prefetching may be triggered by this master 1 Prefetching may be triggered by this master
1 P0_M14PFE	Port0 Master 14 Prefetch enable. This bit controls whether prefetching may be triggered by AHB master 14. 0 No prefetching may be triggered by this master 1 Prefetching may be triggered by this master
2 P0_M13PFE	Port0 Master 13 Prefetch enable. This bit controls whether prefetching may be triggered by AHB master 13. 0 No prefetching may be triggered by this master 1 Prefetching may be triggered by this master
3 P0_M12PFE	Port0 Master 12 Prefetch enable. This bit controls whether prefetching may be triggered by AHB master 12.

Table continues on the next page...

PFLASH_PFCR1 field descriptions (continued)

Field	Description
	0 No prefetching may be triggered by this master 1 Prefetching may be triggered by this master
4 P0_M11PFE	Port0 Master 11 Prefetch enable. This bit controls whether prefetching may be triggered by AHB master 11. 0 No prefetching may be triggered by this master 1 Prefetching may be triggered by this master
5 P0_M10PFE	Port0 Master 10 Prefetch enable. This bit controls whether prefetching may be triggered by AHB master 10. 0 No prefetching may be triggered by this master 1 Prefetching may be triggered by this master
6 P0_M9PFE	Port0 Master 9 Prefetch enable. This bit controls whether prefetching may be triggered by AHB master 9. 0 No prefetching may be triggered by this master 1 Prefetching may be triggered by this master
7 P0_M8PFE	Port0 Master 8 Prefetch enable. This bit controls whether prefetching may be triggered by AHB master 8. 0 No prefetching may be triggered by this master 1 Prefetching may be triggered by this master
8 P0_M7PFE	Port0 Master 7 Prefetch enable. This bit controls whether prefetching may be triggered by AHB master 7. 0 No prefetching may be triggered by this master 1 Prefetching may be triggered by this master
9 P0_M6PFE	Port0 Master 6 Prefetch enable. This bit controls whether prefetching may be triggered by AHB master 6. 0 No prefetching may be triggered by this master 1 Prefetching may be triggered by this master
10 P0_M5PFE	Port0 Master 5 Prefetch enable. This bit controls whether prefetching may be triggered by AHB master 5. 0 No prefetching may be triggered by this master 1 Prefetching may be triggered by this master
11 P0_M4PFE	Port0 Master 4 Prefetch enable. This bit controls whether prefetching may be triggered by AHB master 4. 0 No prefetching may be triggered by this master 1 Prefetching may be triggered by this master
12 P0_M3PFE	Port0 Master 3 Prefetch enable. This bit controls whether prefetching may be triggered by AHB master 3. 0 No prefetching may be triggered by this master 1 Prefetching may be triggered by this master
13 P0_M2PFE	Port0 Master 2 Prefetch enable. This bit controls whether prefetching may be triggered by AHB master 2. 0 No prefetching may be triggered by this master 1 Prefetching may be triggered by this master
14 P0_M1PFE	Port0 Master 1 Prefetch enable. This bit controls whether prefetching may be triggered by AHB master 1. 0 No prefetching may be triggered by this master 1 Prefetching may be triggered by this master

Table continues on the next page...

PFLASH_PFCR1 field descriptions (continued)

Field	Description
15 P0_MOPFE	Port0 Master 0 Prefetch enable. This bit controls whether prefetching may be triggered by AHB master 0. 0 No prefetching may be triggered by this master 1 Prefetching may be triggered by this master
16–18 APC	Address Pipeline Control. This field controls the number of cycles that a subsequent flash read from the opposite AHB port can be initiated prior to the previous read data being valid. This field applies to the configuration of Port0 and Port1. For valid RWSC and APC combinations please refer to the device data sheet. NOTE: 1. A value of '1' in the most significant bit causes the flash controller to function in retrograde/legacy mode, in which there is no pipelining between the sampling of flash read data and the presentation of the next address for flash lookup. 2. Flash operation is not guaranteed for RWSC/APC combinations other than those specified in the data sheet. 000 Pipelined access to the flash disabled. 001 A pipelined access can be initiated 1 cycle before the previous data is valid 010 A pipelined access can be initiated 2 cycles before the previous data is valid 011 A pipelined access can be initiated 3 cycles before the previous data is valid 1xx Pipelined access to the flash is disabled and one wait state is inserted before a subsequent access can be initiated
19 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
20–23 RWSC	Read Wait State Control. This field controls the number of wait-states to be added to the best-case flash array access time for reads. The best-case flash array access time for reads is one cycle. This field must be set to a value corresponding to the operating frequency of the flash memory controller and the actual read access time of the flash memory controller. For valid RWSC and APC combinations please refer to the device data sheet. NOTE: 1. Higher operating frequencies require non-zero settings for this field for proper flash operation. 2. Flash operation is not guaranteed for RWSC/APC combinations other than those specified in the device data sheet. This field applies to the configuration of Port0 and Port1. 0000 No additional wait-states are added 0001 One additional wait-state is added ... 1111 Fifteen additional wait-states are added
24 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
25 P0_DPFEN	Port0 Data Prefetch Enable. This field enables or disables prefetching initiated by a data read access. 0 No prefetching is triggered by a data read access 1 Prefetching may be triggered by any data read access
26 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

Table continues on the next page...

PFLASH_PFCR1 field descriptions (continued)

Field	Description
27 P0_IPFEN	Port0 Instruction Prefetch Enable. This bit enables or disables prefetching initiated by an instruction read access. 0 No prefetching is triggered by an instruction read access 1 Prefetching may be triggered by any instruction read access
28 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
29–30 P0_PFLIM	Port0 PFlash Prefetch Limit. This field controls the prefetch algorithm used by the prefetch controller. This field defines a limit on the maximum number of sequential prefetches which will be attempted between buffer misses. In all situations when enabled, only a single prefetch is initiated on each buffer miss or hit. 00 No prefetching or buffering is performed. 01 The referenced line is prefetched on a buffer miss, that is, prefetch on miss. 10 The referenced line is prefetched on a buffer miss, or the next sequential line is prefetched on a buffer hit (if not already present), that is, prefetch on miss or hit. 11 The referenced line is prefetched on a buffer miss, or the next sequential line is prefetched on a buffer hit (if not already present), that is, prefetch on miss or hit.
31 P0_BFEN	Port0 PFlash Line Read Buffers Enable. This bit enables or disables line read buffer hits. It is also used to invalidate the buffers. This bit can only be updated while PFCR3[BFEN_LK]=0. 0 The line read buffers are disabled from satisfying read requests, and all buffer valid bits are cleared. 1 The line read buffers are enabled to satisfy read requests on hits. Buffer valid bits may be set when the buffers are successfully filled.

25.5.2 Platform Flash Configuration Register 2 (PFLASH_PFCR2)

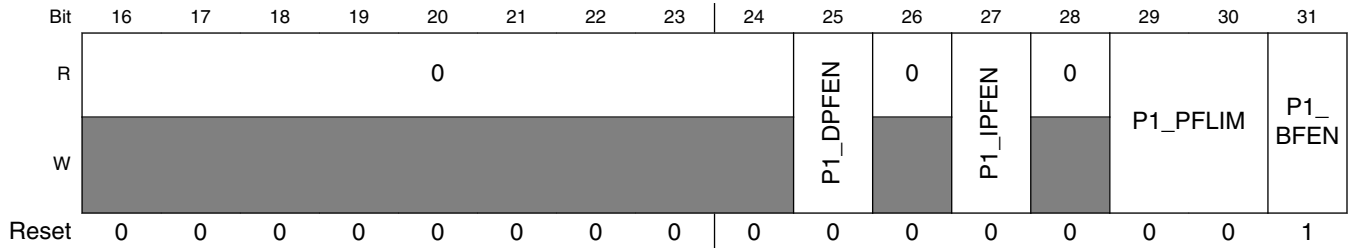
The PFlash Configuration Register 2 (PFCR2) controls the operation of Port1 of the PFLASH_C55FM flash memory controller.

NOTE

See the chip-specific platform flash controller information for details about the masters.

Address: 0h base + 4h offset = 4h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	P1_M15PFE	P1_M14PFE	P1_M13PFE	P1_M12PFE	P1_M11PFE	P1_M10PFE	P1_M9PFE	P1_M8PFE	P1_M7PFE	P1_M6PFE	P1_M5PFE	P1_M4PFE	P1_M3PFE	P1_M2PFE	P1_M1PFE	P1_M0PFE
W	P1_M15PFE	P1_M14PFE	P1_M13PFE	P1_M12PFE	P1_M11PFE	P1_M10PFE	P1_M9PFE	P1_M8PFE	P1_M7PFE	P1_M6PFE	P1_M5PFE	P1_M4PFE	P1_M3PFE	P1_M2PFE	P1_M1PFE	P1_M0PFE
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

**PFLASH_PFCR2 field descriptions**

Field	Description
0 P1_M15PFE	Port1 Master 15 Prefetch enable. This bit controls whether prefetching may be triggered by AHB master 15. 0 No prefetching may be triggered by this master 1 Prefetching may be triggered by this master
1 P1_M14PFE	Port1 Master 14 Prefetch enable. This bit controls whether prefetching may be triggered by AHB master 14. 0 No prefetching may be triggered by this master 1 Prefetching may be triggered by this master
2 P1_M13PFE	Port1 Master 13 Prefetch enable. This bit controls whether prefetching may be triggered by AHB master 13. 0 No prefetching may be triggered by this master 1 Prefetching may be triggered by this master
3 P1_M12PFE	Port1 Master 12 Prefetch enable. This bit controls whether prefetching may be triggered by AHB master 12. 0 No prefetching may be triggered by this master 1 Prefetching may be triggered by this master
4 P1_M11PFE	Port1 Master 11 Prefetch enable. This bit controls whether prefetching may be triggered by AHB master 11. 0 No prefetching may be triggered by this master 1 Prefetching may be triggered by this master
5 P1_M10PFE	Port1 Master 10 Prefetch enable. This bit controls whether prefetching may be triggered by AHB master 10. 0 No prefetching may be triggered by this master 1 Prefetching may be triggered by this master
6 P1_M9PFE	Port1 Master 9 Prefetch enable. This bit controls whether prefetching may be triggered by AHB master 9. 0 No prefetching may be triggered by this master 1 Prefetching may be triggered by this master
7 P1_M8PFE	Port1 Master 8 Prefetch enable. This bit controls whether prefetching may be triggered by AHB master 8. 0 No prefetching may be triggered by this master 1 Prefetching may be triggered by this master
8 P1_M7PFE	Port1 Master 7 Prefetch enable. This bit controls whether prefetching may be triggered by AHB master 7.

Table continues on the next page...

PFLASH_PFCR2 field descriptions (continued)

Field	Description
	0 No prefetching may be triggered by this master 1 Prefetching may be triggered by this master
9 P1_M6PFE	Port1 Master 6 Prefetch enable. This bit controls whether prefetching may be triggered by AHB master 6. 0 No prefetching may be triggered by this master 1 Prefetching may be triggered by this master
10 P1_M5PFE	Port1 Master 5 Prefetch enable. This bit controls whether prefetching may be triggered by AHB master 5. 0 No prefetching may be triggered by this master 1 Prefetching may be triggered by this master
11 P1_M4PFE	Port1 Master 4 Prefetch enable. This bit controls whether prefetching may be triggered by AHB master 4. 0 No prefetching may be triggered by this master 1 Prefetching may be triggered by this master
12 P1_M3PFE	Port1 Master 3 Prefetch enable. This bit controls whether prefetching may be triggered by AHB master 3. 0 No prefetching may be triggered by this master 1 Prefetching may be triggered by this master
13 P1_M2PFE	Port1 Master 2 Prefetch enable. This bit controls whether prefetching may be triggered by AHB master 2. 0 No prefetching may be triggered by this master 1 Prefetching may be triggered by this master
14 P1_M1PFE	Port1 Master 1 Prefetch enable. This bit controls whether prefetching may be triggered by AHB master 1. 0 No prefetching may be triggered by this master 1 Prefetching may be triggered by this master
15 P1_M0PFE	Port1 Master 0 Prefetch enable. This bit controls whether prefetching may be triggered by AHB master 0. 0 No prefetching may be triggered by this master 1 Prefetching may be triggered by this master
16–24 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
25 P1_DPFEN	Port1 Data Prefetch Enable. This field enables or disables prefetching initiated by a data read access. 0 No prefetching is triggered by a data read access 1 Prefetching may be triggered by any data read access
26 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
27 P1_IPFEN	Port1 Instruction Prefetch Enable. This bit enables or disables prefetching initiated by an instruction read access. 0 No prefetching is triggered by an instruction read access 1 Prefetching may be triggered by any instruction read access
28 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
29–30 P1_PFLIM	Port1 PFlash Prefetch Limit. This field controls the prefetch algorithm used by the prefetch controller. This field defines a limit on the maximum number of sequential prefetches which will be attempted between buffer misses. In all situations when enabled, only a single prefetch is initiated on each buffer miss or hit.

Table continues on the next page...

PFLASH_PFCR2 field descriptions (continued)

Field	Description
	00 No prefetching or buffering is performed. 01 The referenced line is prefetched on a buffer miss, that is, prefetch on miss. 10 The referenced line is prefetched on a buffer miss, or the next sequential line is prefetched on a buffer hit (if not already present), that is, prefetch on miss or hit. 11 The referenced line is prefetched on a buffer miss, or the next sequential line is prefetched on a buffer hit (if not already present), that is, prefetch on miss or hit
31 P1_BFEN	Port1 PFlash Line Read Buffers Enable. This bit enables or disables line read buffer hits. It is also used to invalidate the buffers. This bit can only be updated while PFCR3[BFEN_LK]=0. 0 The line read buffers are disabled from satisfying read requests, and all buffer valid bits are cleared. 1 The line read buffers are enabled to satisfy read requests on hits. Buffer valid bits may be set when the buffers are successfully filled.

25.5.3 Platform Flash Configuration Register 3 (PFLASH_PFCR3)

Address: 0h base + 8h offset = 8h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R					0				0			Reserved		0		
W	P0_WCFG		P1_WCFG													BFEN_LK
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R			0						0							
W	ARBM															DERR_SUP
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

PFLASH_PFCR3 field descriptions

Field	Description
0–1 P0_WCFG	Port0 Way Configuration. This field controls the configuration of the line buffers for a given set across all four ways in the controller cache. The indexed set can be organized as a "pool" of available resources, or with a fixed partition between instruction and data buffers. In all cases, when a buffer miss occurs, the flash page is assigned a location within the controller cache. Within the indexed set, the way is selected using a least-recently-used replacement policy, and the entry is then marked as most-recently-used for that set. If the flash access is for the next-sequential line (prefetch), the buffer is not marked as most-recently-used until the given address produces a buffer hit. This field is initialized by hardware reset. 00 All four buffers in an indexed set are available for any flash access, that is, there is no partitioning of the buffers based on the access type. 01 Reserved

Table continues on the next page...

PFLASH_PFCR3 field descriptions (continued)

Field	Description
	10 The buffers in an indexed set are partitioned into two groups with way 0 and 1 allocated for instruction fetches and way 2 and 3 for data accesses. 11 The buffers are partitioned into two groups with way 0, 1, 2 allocated for instruction fetches and way 3 for data accesses.
2–3 P1_WCFG	Port1 Way Configuration. This field controls the configuration of the line buffers for a given set across all four ways in the controller cache. The indexed set can be organized as a "pool" of available resources, or with a fixed partition between instruction and data buffers. In all cases, when a buffer miss occurs, the flash page is assigned a location within the controller cache. Within the indexed set, the way is selected using a least-recently-used replacement policy, and the entry is then marked as most-recently-used for that set. If the flash access is for the next-sequential line (prefetch), the buffer is not marked as most-recently-used until the given address produces a buffer hit. This field is initialized by hardware reset. 00 All four buffers are available for any flash access, that is, there is no partitioning of the buffers based on the access type. 01 Reserved 10 The buffers are partitioned into two groups with ways 0 and 1 allocated for instruction fetches and ways 2 and 3 for data accesses. 11 The buffers are partitioned into two groups with buffers 0,1,2 allocated for instruction fetches and buffer 3 for data accesses.
4–5 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
6–10 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
11 Reserved	This field is reserved.
12–14 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
15 BFEN_LK	BFEN Lock. This field controls access to the PFCRm[Pn_BFEN] fields, which enable or disable line read buffer hits and are also used to invalidate the buffers. After this field is set to 1, it can be cleared to 0 only by a system reset. 0 PFCRm[Pn_BFEN] fields are R/W. 1 PFCRm[Pn_BFEN] fields are Read-only.
16–17 ARBM	Arbitration Mode. This 2-bit field controls the arbitration of concurrent flash access requests from the two AHB ports of the flash memory controller. In both fixed priority or round-robin modes, write requests are prioritized higher than read requests, and read requests are prioritized higher than speculative prefetch requests whenever both ports issue concurrent requests. This field is initialized by hardware reset. 00 Fixed priority arbitration with AHB p0>p1 01 Fixed priority arbitration with AHB p1>p0 10 Round-robin arbitration 11 Round-robin arbitration
18–19 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
20–30 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

Table continues on the next page...

PFLASH_PFCR3 field descriptions (continued)

Field	Description
31 DERR_SUP	Data Error Suppression. Refer to the Embedded Flash Memory configuration information for which Flash blocks are affected by this field. 0 ECC events on data flash accesses are reported. 1 ECC events on data flash accesses are suppressed. Single-bit and multi-bit ECC errors are not reported. On multibit ECC events, the corrupted read data is replaced with a fixed, ECC-clean illegal opcode value of 0x15551555.

25.5.4 Platform Flash Access Protection Register (PFLASH_PFAPR)

The PFlash Access Protection Register (PFAPR) is used to control read and write accesses to the flash array.

NOTE

See the chip-specific platform flash controller information for details about the actual masters available on the device.

Address: 0h base + Ch offset = Ch

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R																
W																
	M0AP	M1AP	M2AP	M3AP	M4AP	M5AP	M6AP	M7AP								
Reset	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R																
W																
	M8AP	M9AP	M10AP	M11AP	M12AP	M13AP	M14AP	M15AP								
Reset	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

PFLASH_PFAPR field descriptions

Field	Description
0–1 M0AP	Master 0 Access Protection. This field controls whether read and write accesses to the flash are allowed based on the master ID of a requesting master. These fields are initialized by hardware reset. 00 No accesses may be performed by this master 01 Only read accesses may be performed by this master 10 Only write accesses may be performed by this master 11 Both read and write accesses may be performed by this master
2–3 M1AP	Master 1 Access Protection. This field controls whether read and write accesses to the flash are allowed based on the master ID of a requesting master. These fields are initialized by hardware reset. 00 No accesses may be performed by this master 01 Only read accesses may be performed by this master 10 Only write accesses may be performed by this master 11 Both read and write accesses may be performed by this master
4–5 M2AP	Master 2 Access Protection. This field controls whether read and write accesses to the flash are allowed based on the master ID of a requesting master. These fields are initialized by hardware reset.

Table continues on the next page...

PFLASH_PFAPR field descriptions (continued)

Field	Description
	00 No accesses may be performed by this master 01 Only read accesses may be performed by this master 10 Only write accesses may be performed by this master 11 Both read and write accesses may be performed by this master
6–7 M3AP	Master 3 Access Protection. This field controls whether read and write accesses to the flash are allowed based on the master ID of a requesting master. These fields are initialized by hardware reset. 00 No accesses may be performed by this master 01 Only read accesses may be performed by this master 10 Only write accesses may be performed by this master 11 Both read and write accesses may be performed by this master
8–9 M4AP	Master 4 Access Protection. This field controls whether read and write accesses to the flash are allowed based on the master ID of a requesting master. These fields are initialized by hardware reset. 00 No accesses may be performed by this master 01 Only read accesses may be performed by this master 10 Only write accesses may be performed by this master 11 Both read and write accesses may be performed by this master
10–11 M5AP	Master 5 Access Protection. This field controls whether read and write accesses to the flash are allowed based on the master ID of a requesting master. These fields are initialized by hardware reset. 00 No accesses may be performed by this master 01 Only read accesses may be performed by this master 10 Only write accesses may be performed by this master 11 Both read and write accesses may be performed by this master
12–13 M6AP	Master 6 Access Protection. This field controls whether read and write accesses to the flash are allowed based on the master ID of a requesting master. These fields are initialized by hardware reset. 00 No accesses may be performed by this master 01 Only read accesses may be performed by this master 10 Only write accesses may be performed by this master 11 Both read and write accesses may be performed by this master
14–15 M7AP	Master 7 Access Protection. This field controls whether read and write accesses to the flash are allowed based on the master ID of a requesting master. These fields are initialized by hardware reset. 00 No accesses may be performed by this master 01 Only read accesses may be performed by this master 10 Only write accesses may be performed by this master 11 Both read and write accesses may be performed by this master
16–17 M8AP	Master 8 Access Protection. This field controls whether read and write accesses to the flash are allowed based on the master ID of a requesting master. These fields are initialized by hardware reset. 00 No accesses may be performed by this master 01 Only read accesses may be performed by this master 10 Only write accesses may be performed by this master 11 Both read and write accesses may be performed by this master
18–19 M9AP	Master 9 Access Protection. This field controls whether read and write accesses to the flash are allowed based on the master ID of a requesting master. These fields are initialized by hardware reset.

Table continues on the next page...

PFLASH_PFAPR field descriptions (continued)

Field	Description
	00 No accesses may be performed by this master 01 Only read accesses may be performed by this master 10 Only write accesses may be performed by this master 11 Both read and write accesses may be performed by this master
20–21 M10AP	Master 10 Access Protection. This field controls whether read and write accesses to the flash are allowed based on the master ID of a requesting master. These fields are initialized by hardware reset. 00 No accesses may be performed by this master 01 Only read accesses may be performed by this master 10 Only write accesses may be performed by this master 11 Both read and write accesses may be performed by this master
22–23 M11AP	Master 11 Access Protection. This field controls whether read and write accesses to the flash are allowed based on the master ID of a requesting master. These fields are initialized by hardware reset. 00 No accesses may be performed by this master 01 Only read accesses may be performed by this master 10 Only write accesses may be performed by this master 11 Both read and write accesses may be performed by this master
24–25 M12AP	Master 12 Access Protection. This field controls whether read and write accesses to the flash are allowed based on the master ID of a requesting master. These fields are initialized by hardware reset. 00 No accesses may be performed by this master 01 Only read accesses may be performed by this master 10 Only write accesses may be performed by this master 11 Both read and write accesses may be performed by this master
26–27 M13AP	Master 13 Access Protection. This field controls whether read and write accesses to the flash are allowed based on the master ID of a requesting master. These fields are initialized by hardware reset. 00 No accesses may be performed by this master 01 Only read accesses may be performed by this master 10 Only write accesses may be performed by this master 11 Both read and write accesses may be performed by this master
28–29 M14AP	Master 14 Access Protection. This field controls whether read and write accesses to the flash are allowed based on the master ID of a requesting master. These fields are initialized by hardware reset. 00 No accesses may be performed by this master 01 Only read accesses may be performed by this master 10 Only write accesses may be performed by this master 11 Both read and write accesses may be performed by this master
30–31 M15AP	Master 15 Access Protection. This field controls whether read and write accesses to the flash are allowed based on the master ID of a requesting master. These fields are initialized by hardware reset. 00 No accesses may be performed by this master 01 Only read accesses may be performed by this master 10 Only write accesses may be performed by this master 11 Both read and write accesses may be performed by this master

25.6 Functional description

The flash memory controller interfaces between:

- The AHB system bus ports
- Flash memory array

For accesses targeting the flash array, the flash memory controller generates read and write enables, block selects, array address, write size and write data as inputs to the flash array. The flash memory controller captures read data from the flash array and drives it onto the AHB system bus. Up to four pages of data (256-bit page size) may be buffered in each of four ways of the flash memory controller mini-cache. Lines may be prefetched in advance of being requested, allowing single-cycle (zero AHB wait-states) read data responses on buffer hits.

Several prefetch control algorithms are available for controlling line read buffer fills. Prefetch triggering may be restricted to instruction accesses only, data accesses only, or may be unrestricted. Prefetch triggering may also be controlled on a per-master basis.

Buffers may also be selectively enabled or disabled for allocation by instruction and data prefetch.

Access protections may be applied on a per-master basis for both reads and writes to support security and privilege mechanisms.

25.6.1 Basic interface protocol

Read accesses are terminated under control of the appropriate wait state settings. Thus, the access time of the operation is determined by the setting of the PFCR1[RWSC] field. Access timing can be varied to account for the operating conditions of the SoC (frequency, voltage, temp) by appropriately setting the PFCR1[RWSC] field.

25.6.2 Access protections

On-chip Flash memory accesses are subject to restrictions based on chip security implementation and/or requirements based on functional restrictions of the flash memory, e.g., a read access to a flash block while it is currently being written is not allowed. When a prohibited access is attempted, a system bus error termination results.

The flash memory controller may invoke a system bus error termination in the following scenarios:

- Attempted access by an AHB master whose corresponding read access control or write access control settings do not allow the access, thus causing a protection violation. See [Platform Flash Access Protection Register \(PFLASH_PFAPR\)](#) for more detail. In this case the flash memory controller does not initiate the requested flash array access.
- The flash returns an error response on an attempted access to a partition that is unavailable. (see [Flash error response operation](#) or [Embedded Flash Memory](#)).
- Attempted access by an AHB master to a reserved region in the flash memory map.

The flash array may also signal an error response to terminate a requested access due to improper sequencing during program/erase operations, improper sequencing during array integrity testing. When an error response is received the flash memory controller does not update or validate a page read buffer. An error response may be signaled on a read or interlock write operation. For more information on the specifics related to signaling of flash errors, including flash ECC events, array integrity testing and read-while-write events, refer to the flash memory chapter.

25.6.2.1 PFAPR - Platform Flash Access Protection Register (PFAPR)

The flash memory controller provides programmable, configurable access protections for both read and write cycles on a per-master basis via the PFlash Access Protection Register (PFAPR). It allows restriction of read and write requests on a per-master basis. This functionality is described in [Platform Flash Access Protection Register \(PFLASH_PFAPR\)](#). Detection of a protection violation results in an error response from the flash memory controller on the AHB transfer.

25.6.3 Access pipelining

Accesses to the flash array may be pipelined by driving a subsequent access address and control signals while waiting for the current access to complete. Pipelined access requests are always run to completion and are not aborted by the flash memory controller. Flash access pipelining allows for improved performance by reducing the access latency seen by the AHB master. Access pipelining may be applied only to read cycles targeting the flash array. Address pipelining is enabled by setting the PFCR1[APC] field.

Access pipelining is only supported for normal flash access .

NOTE

Not all combinations of PFCR1[APC] (Address Pipeline Control) and PFCR1[RWSC] (Read Wait State Control)

settings are recommended or supported. Please refer to the device data sheet for guidance.

25.6.4 Line read buffers and prefetch operation

The AHB ports of the flash memory controller each contain a four-way set-associative mini cache, where each way contains four page buffers which are used to hold data read from the flash arrays. Each 256-bit buffer operates independently, and is filled using a single array access. The buffers are used for both prefetch and normal demand fetches.

Prefetch triggering is controllable on a per-master and access-type basis (see [Platform Flash Configuration Register 1 \(PFLASH_PFCR1\)](#)). Bus masters may be enabled or disabled from triggering prefetches, and triggering may be further restricted based on whether a read access is for instruction or data. A read access by the flash memory controller may trigger a prefetch to the next sequential line of array data on the cycle following the request. The access address is incremented by 32-bytes, and a subsequent flash access is initiated. A flash array prefetch is initiated if the data is not already resident in a line read buffer. Prefetched data is always loaded into the least-recently-used buffer.

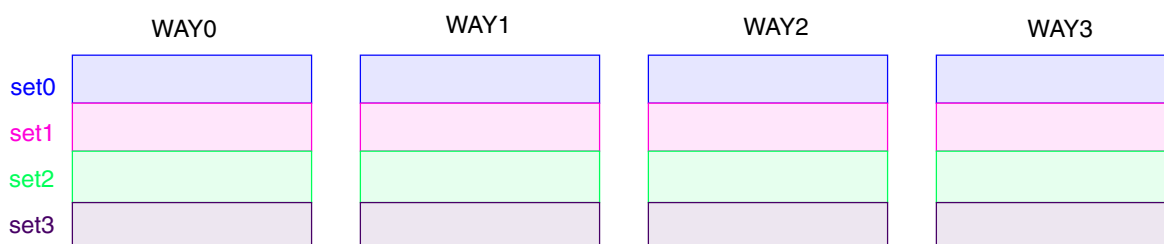


Figure 25-6. Flash memory controller 4-entry, 4-way mini-cache organization

Once the candidate line buffer has been selected, the flash array is accessed and read data loaded into the buffer. If the buffer load was in response to a miss, the buffer which was loaded is immediately marked as most-recently-used. If the buffer load was in response to a speculative fetch to the next-sequential line address after a buffer hit, *the recently-used status is not changed*. Rather, it is marked as most-recently-used only after a subsequent buffer hit.

This policy maximizes performance based on reference patterns of flash accesses and allows for prefetched data to remain valid when non-prefetch enabled bus masters are granted flash access.

Several algorithms are available for prefetch control which trade off performance for power. They are described in [Platform Flash Configuration Register 1 \(PFLASH_PFCR1\)](#). More aggressive prefetching may increase power due to the number of potentially discarded prefetches, but may increase performance by lowering average read latency.

For prefetching to occur, all of the following must apply:

1. PFCR{1,2}[P{0,1}_BFEN] must be set to '1',
2. PFCR{1,2}[P{0,1}_PFLIM] must be non-zero, and
3. Either PFCR{1,2}[P{0,1}_IPFEN] or PFCR{1,2}[P{0,1}_DPFEN] must be asserted.

Refer to [Platform Flash Configuration Register 1 \(PFLASH_PFCR1\)](#) for a description of these controls.

25.6.5 Instruction/Data prefetch triggering

Port0 prefetch triggering may be enabled for instruction reads via the PFCR1[IPFEN] control field, while Port0 prefetching for data reads is enabled via the PFCR1[DPFEN] control field. Additionally, the PFCR1[PFLIM] must also be set to enable prefetching on Port0. Refer to [Platform Flash Configuration Register 1 \(PFLASH_PFCR1\)](#) for a description of these controls. Prefetches are never triggered by write cycles.

Port1 prefetch triggering may be enabled for instruction reads via the PFCR2[IPFEN] control field, while Port1 prefetching for data reads is enabled via the PFCR2[DPFEN] control field. Additionally, the PFCR2[PFLIM] must also be set to enable prefetching on Port1. Refer to [Platform Flash Configuration Register 2 \(PFLASH_PFCR2\)](#) for a description of these controls. Prefetches are never triggered by write cycles.

25.6.6 Per-Master prefetch triggering

Prefetch triggering may be controlled for individual bus masters. Refer to [Platform Flash Configuration Register 1 \(PFLASH_PFCR1\)](#) for a description of these controls.

25.6.7 Buffer allocation

Allocation of the line read buffers is controlled via the PFCR3 control register, specifically the line buffer configuration ({P0,P1}_WCFG) field. Refer to [Platform Flash Configuration Register 3 \(PFLASH_PFCR3\)](#). The buffers can be organized as a "pool" of available resources (with all four ways within a given set) or with a fixed partition between ways allocated to instruction or data accesses. For the fixed partitions, two

configurations are supported. In one configuration, ways 0 and 1 are allocated for instruction fetches and ways 2 and 3 for data accesses. In the second configuration, ways 0, 1 and 2 are allocated for instruction fetches and way 3 reserved for data accesses.

25.6.8 Reliability considerations

This section summarizes mechanism to enhance the reliability of flash by correcting and detecting faults.

25.6.8.1 e2eECC End-to-End ECC

25.6.8.1.1 e2eECC and flash accesses

There are multiple complications associated with the use of the standard e2eECC algorithm with flash memory. The basic issues involve: the default state of a flash block that is erased (all ones) and the fact that programming an erased flash involves changing the state of memory bit from a logical 1 to a logical 0. These factors require that the system level e2eECC must be modified on references to the flash memory in three ways.

First, the all-ones codeword, since it reflects the state of an erased flash location, is treated as a valid error free encoding; in particular, it is required that the checkbits associated with the all all-ones value and the all zeroes data value are required to be the same with a checkbit value of 0xFF. Second, since an erased flash block generates an all-ones data for all locations, the address field cannot be included in the ECC code used by the flash. The flash array implements special address re-encoding logic based on the decoded array address to protect against address faults. Third, there are additional ECC implications associated with the flash memory and its support for EEPROM emulation.

In summary, the basic operation of an NVM bit cell impacts the e2eECC algorithm used in the flash memory in multiple ways. The required ECC adjustments are handled by the platform flash controller before data is written to the flash array(s) or applied to read data accessed from the array(s). These adjustments are two-fold. Consider a flash write during a programming event:

1. Remove the address field from the ECC codeword by XOR'ing the address calculation of the H-matrix from the checkbits.
2. Invert the resulting 8-bit ECC checkbit vector. This step is required to support the all-ones erased state.

On a flash read operation, a similar but "reversed" set of steps are performed as the data is driven into the system bus interconnect:

1. Invert the 8-bit ECC checkbit vector read from the flash.
2. Factor the address field into the ECC checkbits by XOR'ing the address calculation of the H-matrix.

Consider the following flash ECC example. Let the flash block containing address 0x00345678 be erased. The following is the sequence of operations:

1. Read address 0x00345678. Flash array returns `fl_rdata = 0xFFFFFFFF_FFFFFFFF`, `fl_cdata = 0xFF`. The flash memory controller calculates the `addr_chkbit = 0xC4` using the H-matrix; it inverts the `fl_cdata` vector to 0x00 and factors in the `addr_chkbit = 0xC4` to produce the following valid e2eECC codeword:

`addr = 0x00345678`, `rdata = 0xFFFFFFFF_FFFFFFFF`, `rchkbit = 0xC4` ($0x00 \wedge 0xC4$)

2. Program address 0x00345678 with data = 0xBABEFACE_DEADBEEF. The initiating bus master calculates the e2eECC codeword as:

`addr = 0x00345678`, `wdata = 0xBABEFACE_DEADBEEF`, `wchkbit = 0xE3`

Before performing the interlock write to the flash array, the flash memory controller adjusts the `chkbit` value as `fl_wchkbit = 0xD8` ($0xE3 \wedge 0xFF \wedge 0xC4$) by toggling the original write checkbits and then factoring in the `addr_ecc`. Accordingly, the codeword stored in the flash array is:

`fl_data = 0xBABEFACE_DEADBEEF`, `fl_checkbit = 0xD8`

3. Read address 0x00345678. The flash array returns the stored codeword, `fl_rdata = 0xBABEFACE_DEADBEEF`, `fl_rchkbit = 0xD8`. The flash memory controller inverts the checkbit vector and factors in the address checkbits ($0xD8 \wedge 0xFF \wedge 0xC4$) to produce:

`addr = 0x00345678`, `rdata = 0xBABEFACE_DEADBEEF`, `rchkbit = 0xE3`

This read codeword matches the original e2eECC codeword generated by the write.

Malfunction of the ECC logic described above may result in the corruption of the SEC/DED event reporting. The flash memory controller performs EDC after ECC check on all flash transactions. If a mismatch is detected, indicating a failure in the ECC logic, the event is reported to the device fault collection module.

25.6.8.2 Flash address generation check

Fault detection coverage of the address and data are handled by ECC performed within the flash and e2eECC performed at the master. Fault detection coverage of the address path and control within the flash memory controller rely on a feedback path between the flash controller and flash. Recall on a requested access to flash, the flash memory controller must decode the system AHB bus signals to generate the corresponding flash interface signals to invoke a flash lookup. In addition to providing the requested read data, the flash also provides output sidebands reflecting the encoded address and block selects used to perform the actual row lookup.

This sideband information is used by the flash memory controller to verify the expected transaction. If a mismatch is detected, indicating a failure in the address generation or control logic within the flash memory controller or the transmission path between the flash memory controller and flash array, the event is forwarded to the device fault collection module and the corresponding buffer is invalidated.

25.6.9 ECC on data flash accesses

For single-bit correction events, the corrected data and checkbits are returned to the requesting master.

In the event of a non-correctable error detection, a fixed, illegal opcode value (1555_1555h) is returned to the requesting master along with the associated ECC checkbits as determined by the requesting address.

When PFCR3[DERR_SUP] is enabled, ECC errors on data flash blocks are suppressed from being reported.

- In the event of a single-bit correction, the corrected data and checkbits are returned to the requesting master, and the single-bit correction event is suppressed from being reported.
- In the event of a non-correctable error detection, a fixed, illegal opcode value (1555_1555h) is returned to the requesting master along with the associated ECC checkbits as determined by the requesting address and the non-correctable error event is suppressed from being reported.

NOTE

EEPROM should be avoided for storage of executable code.

25.6.10 Array integrity considerations

During an array integrity sequence, the flash memory array ignores any incoming read requests. When a flash array integrity check is in progress, the flash memory controller terminates all flash access requests with an error. More specifically, it aborts the incoming flash access requests and terminates the system bus transfer with an error.

Chapter 26

Embedded Flash Memory (C55FMC)

26.1 Chip-specific C55FMC information

NOTE

In production devices of this chip, the one-time-programmable Test Mode Disable Seal is programmed—that is, UTest is sealed—at the factory. For more information about the seal, see [Test mode disable seal](#).

26.1.1 C55FMC register reset values

The following table lists C55FMC register reset values that are specific to this chip. For the reset values of all other C55FMC registers, see [Memory map and register definition](#).

Table 26-1. C55FMC register reset values

Register	Reset value
Extended Module Configuration Register (C55FMC_MCRE)	1001_2020h
Over-Program Protection 0 register (C55FMC_OPP0)	Dependent on customer settings ¹
Over-Program Protection 1 register (C55FMC_OPP1)	
Over-Program Protection 2 register (C55FMC_OPP2)	
Over-Program Protection 3 register (C55FMC_OPP3)	

1. The reset value of the C55FMC_OPPn registers, for each flash memory block mapped to a bit within each register, reflects whether the block has been marked as One Time Programmable (OTP). The C55FMC_OPPn registers have the same bit-to-block mapping as the C55FMC_LOCKn registers. A value of 1 indicates a block has been marked as OTP. The reset values of the C55FMC_OPPn are similar to the reset values of the TDM_OTPENn registers in the Tamper Detection Module (TDM).

26.1.2 Access to and configuration of blocks mapped to C55FMC_LOCK1 register

The CSE has exclusive access to the specific flash memory blocks that are mapped to the C55FMC_LOCK1 register. Other masters on the chip cannot access these particular blocks.

The only C55FMC register that configures the use of the CSE-exclusive blocks is the OPP1 register. The LOCK1 and SEL1 registers do not configure these blocks.

26.1.3 C55FMC_LOCK n register bit mapping

In the main program and erase interface, the C55FMC_LOCK n registers provide a means to protect flash memory blocks from being modified. Each implemented bit in a C55FMC_LOCK n register maps to a specific block.

26.1.3.1 C55FMC_LOCK0 register bit mapping

The flash memory blocks mapped to the [Lock 0 register \(C55FMC_LOCK0\)](#) bits are as follows.

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
Register Field	TSLOCK	Reserved	LOWLOCK[13:0]														MIDLOCK[15:0]															
Flash Memory Block Name	UTest NVM Block Space 16 KB	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	
	64 KB EEPROM Data Block1 - Partition 1																															
	64 KB EEPROM Data Block0 - Partition 0																															
		—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	
		—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	
		—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	
		—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	
		—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	
		—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	
		—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	
		—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	
		—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	
		—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	
		—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	
		—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	
		—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	
		—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	
		—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	
		—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	
		—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	
		—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	
		—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	
		—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	
		—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	
		—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	
		—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	
		—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	
		—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	
		—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	
		—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	
		—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	
		—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	
		—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	
		—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	
		—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	
		—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	
		—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	
		—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	
		—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	
		—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	
		—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	
		—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	
		—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	
		—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	
		—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	
		—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	
		—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	
		—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	
		—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	
		—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	
		—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	
		—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	
		—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	
		—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	
		—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	
		—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	
		—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	
		—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	
		—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	
		—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	
		—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—			

26.1.3.2 C55FMC_LOCK1 register bit mapping

The flash memory blocks mapped to the [Lock 1 register \(C55FMC_LOCK1\)](#) bits are as follows.

NOTE

This register cannot be used to manage the blocks mapped to the register's implemented bits because the CSE module has exclusive access to these blocks. The bits' value is 1 to reflect the corresponding blocks' locked state, and the bits cannot be written to change that state.

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
Register Field	Reserved																HIGHLOCK[15:0]															
Flash Memory Block Name																															16 KB CSE Data Block1 - Partition 5	16 KB CSE Data Block0 - Partition 4

26.1.3.3 C55FMC_LOCK2 register bit mapping

The flash memory blocks mapped to the [Lock 2 register \(C55FMC_LOCK2\)](#) bits are as follows.

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31

Table continues on the next page...

Flash Memory Block Name	Register Field
256 KB Application Code Block31 - Partition 9	A256KLOCK[31:0]
256 KB Application Code Block30 - Partition 9	
256 KB Application Code Block29 - Partition 9	
256 KB Application Code Block28 - Partition 9	
256 KB Application Code Block27 - Partition 9	
256 KB Application Code Block26 - Partition 9	
256 KB Application Code Block25 - Partition 9	
256 KB Application Code Block24 - Partition 9	
256 KB Application Code Block23 - Partition 8	
256 KB Application Code Block22 - Partition 8	
256 KB Application Code Block21 - Partition 8	
256 KB Application Code Block20 - Partition 8	
256 KB Application Code Block19 - Partition 8	
256 KB Application Code Block18 - Partition 8	
256 KB Application Code Block17 - Partition 8	
256 KB Application Code Block16 - Partition 8	
256 KB Application Code Block15 - Partition 7	
256 KB Application Code Block14 - Partition 7	
256 KB Application Code Block13 - Partition 7	
256 KB Application Code Block12 - Partition 7	
256 KB Application Code Block11 - Partition 7	
256 KB Application Code Block10 - Partition 7	
256 KB Application Code Block9 - Partition 7	
256 KB Application Code Block8 - Partition 7	
256 KB Application Code Block7 - Partition 6	
256 KB Application Code Block6 - Partition 6	
256 KB Application Code Block5 - Partition 6	
Calibration Block4 - Partition 6	
Calibration Block3 - Partition 6	
Calibration Block2 - Partition 6	
Calibration Block1 - Partition 6	
Boot Code Block0 - Partition 6	

26.1.3.4 C55FMC_LOCK3 register bit mapping

The flash memory blocks mapped to the [Lock 3 register \(C55FMC_LOCK3\)](#) bits are as follows.

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
Register Field	Reserved															A256KLOCK[15:0]																

Table continues on the next page...

[illegible]

26.1.4 C55FMC_SEL n register bit mapping

The C55FMC_SEL*n* registers provide a means to select flash memory blocks to be erased. Each implemented bit in a C55FMC_SEL*n* register maps to a specific memory block.

26.1.4.1 C55FMC_SEL0 register bit mapping

The flash memory blocks mapped to the [Select 0 register \(C55FMC_SEL0\)](#) bits are as follows.

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
Register Field	Reserved	LOWSEL[13:0]													MIDSEL[15:0]																	

Table continues on the next page...

Flash Memory Block Name
—
—
—
—
—
—
—
—
—
—
—
—
—
64 KB EEPROM Data Block1 - Partition 1
64 KB EEPROM Data Block0 - Partition 0
—
—
—
—
—
—
—
—
—
—
—
—
64 KB EEPROM Data Block1 - Partition 3
64 KB EEPROM Data Block0 - Partition 2

26.1.4.2 C55FMC_SEL1 register bit mapping

The flash memory blocks mapped to the [Select 1 register \(C55FMC_SEL1\)](#) bits are as follows.

NOTE

This register cannot be used to select the blocks mapped to the register's implemented bits because the CSE module has exclusive access to the blocks.

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
Register Field	Reserved															HIGHSEL[15:0]																

Table continues on the next page...

[illegible]

26.1.4.3 C55FMC_SEL2 register bit mapping

The flash memory blocks mapped to the [Select 2 register \(C55FMC_SEL2\)](#) bits are as follows.

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
Register Field	A256KSEL[31:0]																															

Table continues on the next page...

Flash Memory Block Name
256 KB Application Code Block31 - Partition 9
256 KB Application Code Block30 - Partition 9
256 KB Application Code Block29 - Partition 9
256 KB Application Code Block28 - Partition 9
256 KB Application Code Block27 - Partition 9
256 KB Application Code Block26 - Partition 9
256 KB Application Code Block25 - Partition 9
256 KB Application Code Block24 - Partition 9
256 KB Application Code Block23 - Partition 8
256 KB Application Code Block22 - Partition 8
256 KB Application Code Block21 - Partition 8
256 KB Application Code Block20 - Partition 8
256 KB Application Code Block19 - Partition 8
256 KB Application Code Block18 - Partition 8
256 KB Application Code Block17 - Partition 8
256 KB Application Code Block16 - Partition 8
256 KB Application Code Block15 - Partition 7
256 KB Application Code Block14 - Partition 7
256 KB Application Code Block13 - Partition 7
256 KB Application Code Block12 - Partition 7
256 KB Application Code Block11 - Partition 7
256 KB Application Code Block10 - Partition 7
256 KB Application Code Block9 - Partition 7
256 KB Application Code Block8 - Partition 7
256 KB Application Code Block7 - Partition 6
256 KB Application Code Block6 - Partition 6
256 KB Application Code Block5 - Partition 6
Calibration Block4 - Partition 6
Calibration Block3 - Partition 6
Calibration Block2 - Partition 6
Calibration Block1 - Partition 6
Boot Code Block0 - Partition 6

26.1.4.4 C55FMC_SEL3 register bit mapping

The flash memory blocks mapped to the [Select 3 register \(C55FMC_SEL3\)](#) bits are as follows.

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
Register Field	Reserved															A256KSEL[15:0]																
Flash Memory Block Name																																

26.1.5 C55FMC_OPP n register bit mapping

The C55FMC_OPP n registers provide a means to protect blocks from being over-programmed. Each implemented bit in a C55FMC_OPP n register maps to a specific memory block.

26.1.5.1 C55FMC_OPP0 register bit mapping

The flash memory blocks mapped to the [Over-Program Protection 0 register \(C55FMC_OPP0\)](#) bits are as follows.

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
Register Field	Reserved	Reserved	LOWOPP[13:0]														MIDOPP[15:0]															
Flash Memory Block Name	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	64 KB EEPROM Data Block1 - Partition 1	64 KB EEPROM Data Block0 - Partition 0	—	—	—	—	—	—	—	—	—	—	—	—	—	64 KB EEPROM Data Block1 - Partition 3	64 KB EEPROM Data Block0 - Partition 2

26.1.5.2 C55FMC_OPP1 register bit mapping

The flash memory blocks mapped to the [Over-Program Protection 1 register \(C55FMC_OPP1\)](#) bits are as follows.

NOTE

When the value of a bit in this register reflects that the corresponding flash block has over-program protection, the

Chip-specific C55FMC information

protection applies to the block even though the CSE module has exclusive access to the block.

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	
Register Field	Reserved															HIGHOPP[15:0]																	
Flash Memory Block Name	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	16 KB CSE Data Block1 - Partition 5	16 KB CSE Data Block0 - Partition 4

26.1.5.3 C55FMC_OPP2 register bit mapping

The flash memory blocks mapped to the [Over-Program Protection 2 register \(C55FMC_OPP2\)](#) bits are as follows.

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
Register Field	A256KOPP[31:0]																															

Table continues on the next page...

Flash Memory Block Name
256 KB Application Code Block31 - Partition 9
256 KB Application Code Block30 - Partition 9
256 KB Application Code Block29 - Partition 9
256 KB Application Code Block28 - Partition 9
256 KB Application Code Block27 - Partition 9
256 KB Application Code Block26 - Partition 9
256 KB Application Code Block25 - Partition 9
256 KB Application Code Block24 - Partition 9
256 KB Application Code Block23 - Partition 8
256 KB Application Code Block22 - Partition 8
256 KB Application Code Block21 - Partition 8
256 KB Application Code Block20 - Partition 8
256 KB Application Code Block19 - Partition 8
256 KB Application Code Block18 - Partition 8
256 KB Application Code Block17 - Partition 8
256 KB Application Code Block16 - Partition 8
256 KB Application Code Block15 - Partition 7
256 KB Application Code Block14 - Partition 7
256 KB Application Code Block13 - Partition 7
256 KB Application Code Block12 - Partition 7
256 KB Application Code Block11 - Partition 7
256 KB Application Code Block10 - Partition 7
256 KB Application Code Block9 - Partition 7
256 KB Application Code Block8 - Partition 7
256 KB Application Code Block7 - Partition 6
256 KB Application Code Block6 - Partition 6
256 KB Application Code Block5 - Partition 6
Calibration Block4 - Partition 6
Calibration Block3 - Partition 6
Calibration Block2 - Partition 6
Calibration Block1 - Partition 6
Boot Code Block0 - Partition 6

26.1.5.4 C55FMC_OPP3 register bit mapping

The flash memory blocks mapped to the [Over-Program Protection 3 register \(C55FMC_OPP3\)](#) bits are as follows.

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
Register Field	Reserved															A256KOPP[15:0]																
Flash Memory Block Name																																

26.2 Introduction

The primary function of the embedded flash memory is to serve as electrically programmable and erasable non-volatile memory (NVM) that may be used for instruction or data storage.

The following figure shows the top-level diagram and functional organization of the flash memory unit.

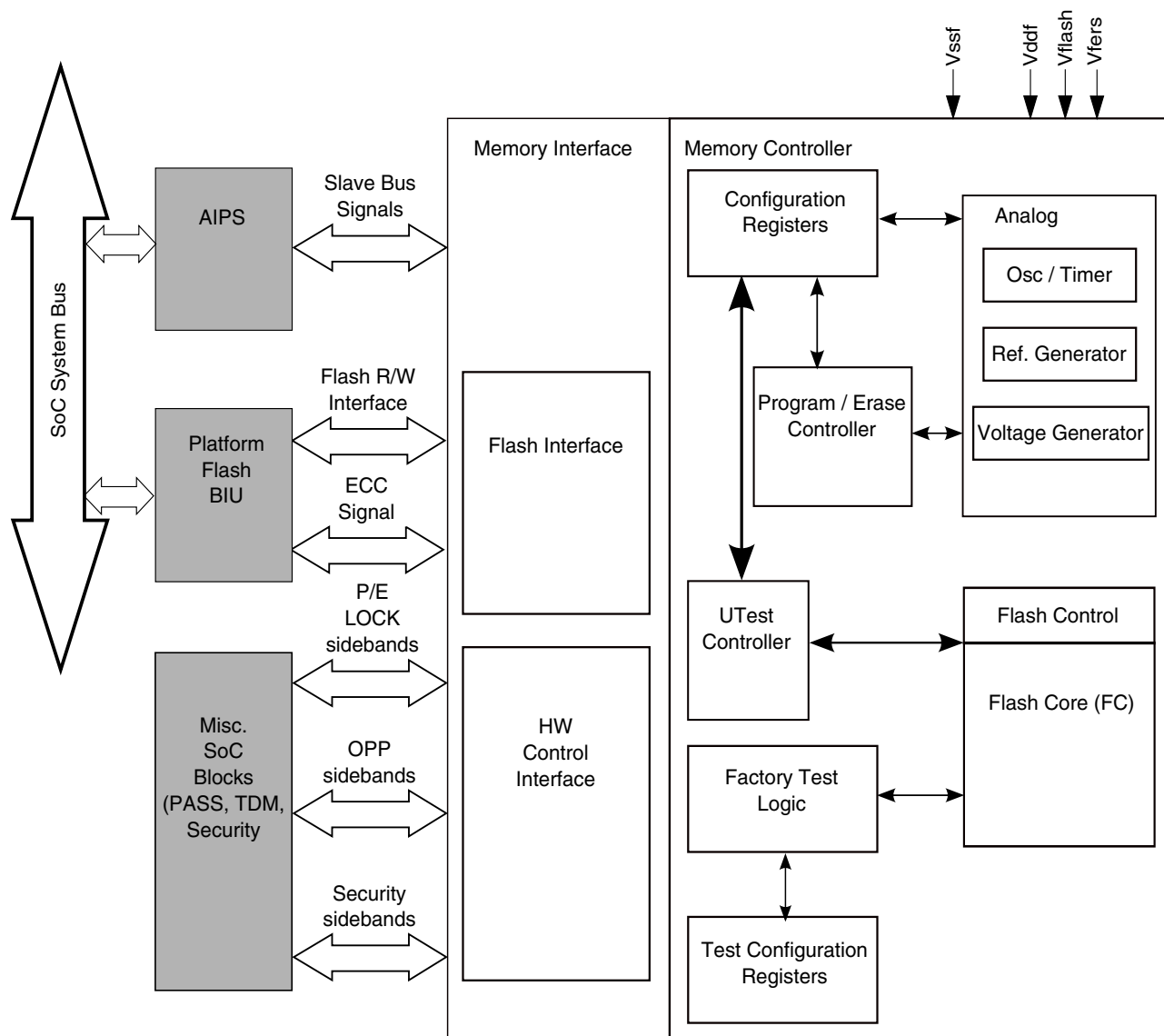


Figure 26-1. c55fmc block diagram

26.2.1 Overview

The embedded flash memory is addressable by word (32 bits) or double word (64 bits) for program operation, and page (256 bits) for read operation. Multiple word or double-word writes may be done to the flash memory to fill up the program page buffer (256 bits), enabling page programming (256 bits, requiring 4 double-word writes) and quad-page programming (1024 bits, requiring 16 double-word writes). Flash memory reads always return 256 bits, although read page buffering may be performed by the Bus Interface Unit (BIU).

For more details on the embedded flash memory architecture and features, see [Functional Description](#).

26.2.2 Features

The embedded flash memory includes these distinct features:

- Test information stored in a dedicated nonvolatile block (referred to as the UTest block)
- OTP space made available in the UTest block
- Read page size of 256 bits (8 words)
- ECC with single-bit correction, double-bit detection (all 1's valid)
- Quad Page programming (64-bit granularity)
- Software programmable block program/erase restriction control
- Erasing of selected block(s)
- Independent programming of the UTest NVM block
- Embedded hardware program and erase algorithms
- Support for reading while writing when the accesses are to different partitions
- Erase suspend, program suspend, and erase-suspended program
- UTest mode (user-accessible test modes) including Array Integrity and Margin Read
- Triple Voted Flops for flash functions requiring high reliability, e.g., internal trimming, redundancy, and mode control

26.2.3 Modes of operation

Following is a brief description of the embedded flash memory operating modes.

- *User mode* is the default operating mode of the embedded flash memory. In this mode, it is possible to read and write registers, read and interlock write the memory

array, program the memory array, and erase the memory array. In this mode program and erase operations are initiated by doing array and register writes, and are controlled by an internal state machine.

- *UTest mode* is a tiered test mode strategy in which a portion of the factory test modes are made available. This mode is protected but accessible.

26.3 UTest NVM block

The UTest NVM block may be enabled by the BIU. When the UTest NVM space is enabled, all operations are mapped to the UTest NVM block. User-mode programming of the UTest NVM block is enabled only when MCR[PEAS] is high. The UTest NVM block is an OTP block (assuming Test Mode Disable Seal is written) - thus erase is not allowed.

The UTest NVM block supports RWW, and is grouped with the blocks in partition 0.

The UTest NVM block may be locked against program by using the lock registers.

Programming of the UTest NVM space has restrictions that are similar to those of the main space in terms of how ECC is calculated. Only one program operation is allowed per 64-bit ECC segment.

The UTest NVM block contains specified data that is needed for embedded flash memory or SoC features.

26.4 Test mode disable seal

Also included in the UTest NVM block is a mechanism to disable factory entry into Test mode. Extreme care must be taken when using this feature, as blocks that are selected to be protected in this method will not be able to have possible failures analyzed by factory failure analysts. Once sealed, the UTest block becomes OTP (Erase Locked, and Over Program Protection enabled). The UTest Block also is not accessible in Test mode.

Protection of this sort prevents all high-voltage operations to the flash executed by the internal state-machine, as well as reads through this state machine, and reads through the Array Integrity state machine when using Test mode interfaces.

UTest operations Margin Read and Array Integrity are also protected by preventing MISR updates on blocks selected for protection, although reads will still be done and single bit corrections and double bit detections will be logged. Protection through other interfaces is protected by normal User mode protection mechanisms, and are device-specific.

The method to disable factory entry into Test mode is to first program the Test Mode Disable Seal location to be 0x2D3C_4B5A. Once the next reset is asserted, Test mode will be disabled.

It is possible to create a password to enable factory entry into test mode. This can be programmed into the Test Mode Disable Override Passcode. The passcode may not be 0x0000_0000, 0xFFFF_FFFF, or 0x5555_5555. These are all invalid passcodes and will not be accepted to override. If it is desired by the customer that override never be possible, one of the three invalid passcodes should be put into this location. Passcode may be entered to authenticate entry (if enabled) by doing a 32-bit register write to register address 0x90. The UTest NVM block is always protected even if the Test Mode Disable Seal password is written. UTest operations Margin Read and Array Integrity are always protected even if the Test Mode Disable Seal password is written.

Only blocks selected in the Test Mode Disable Block Select field(s) are controlled by the Test Mode Disable feature. Thus it is possible for customers to selectively pick blocks that have this type of protection, and will not be eligible for factory failure analysis. Bits programmed to 0 in the Test Mode Disable Block Select field(s) will designate blocks that are controlled by the Test Mode Disable feature. If either Test Mode Disable Block Select - 0 or Test Mode Disable Block Select - 1 have a 0 programmed, that block will be protected. In order to provide two opportunities to select blocks for the Test Mode Disable, two regions are available, Test Mode Disable Block Select - 0 and Test Mode Disable Block Select - 1. The bit of these two regions are logically ANDed, to define which blocks are effectively selected for the Test Mode Disable feature. The Block Select field is organized as follows, and aligns with how blocks are defined in the LOCK registers.

Table 26-2. Test mode disable block select

Block	Bits used to select blocks
Low Block Disable	Data[13:0]
Unused	Data[15:14]
Mid Block Disable	Data[31:16]
High Block Disable	Data[47:32]
256K Block Disable	Data[95:48]

26.5 Memory map and register definition

The embedded flash memory map consists of a flash memory array (which includes main array space and UTest NVM space) and a region of registers associated with the programming model that enable flash memory array operation and modification.

The address space consists of 16 KB, 32 KB, 64 KB, and 256 KB blocks.

C55FMC memory map

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
0	Module Configuration Register (C55FMC_MCR)	32	R/W	0000_0600h	26.5.1/918
8	Extended Module Configuration Register (C55FMC_MCRE)	32	R	See section	26.5.2/923
10	Lock 0 register (C55FMC_LOCK0)	32	R/W	BFFF_FFFFh	26.5.3/926
14	Lock 1 register (C55FMC_LOCK1)	32	R/W	0000_FFFFh	26.5.4/928
18	Lock 2 register (C55FMC_LOCK2)	32	R/W	FFFF_FFFFh	26.5.5/928
1C	Lock 3 register (C55FMC_LOCK3)	32	R/W	0000_FFFFh	26.5.6/929
38	Select 0 register (C55FMC_SEL0)	32	R/W	0000_0000h	26.5.7/930
3C	Select 1 register (C55FMC_SEL1)	32	R/W	0000_0000h	26.5.8/931
40	Select 2 register (C55FMC_SEL2)	32	R/W	0000_0000h	26.5.9/932
44	Select 3 register (C55FMC_SEL3)	32	R/W	0000_0000h	26.5.10/933
50	Address register (C55FMC_ADR)	32	R	0000_0000h	26.5.11/934
54	UTest 0 register (C55FMC_UT0)	32	R/W	0000_0001h	26.5.12/936
58	UMISR register (C55FMC_UM0)	32	R/W	0000_0000h	26.5.13/939
5C	UMISR register (C55FMC_UM1)	32	R/W	0000_0000h	26.5.13/939
60	UMISR register (C55FMC_UM2)	32	R/W	0000_0000h	26.5.13/939
64	UMISR register (C55FMC_UM3)	32	R/W	0000_0000h	26.5.13/939
68	UMISR register (C55FMC_UM4)	32	R/W	0000_0000h	26.5.13/939
6C	UMISR register (C55FMC_UM5)	32	R/W	0000_0000h	26.5.13/939
70	UMISR register (C55FMC_UM6)	32	R/W	0000_0000h	26.5.13/939
74	UMISR register (C55FMC_UM7)	32	R/W	0000_0000h	26.5.13/939

Table continues on the next page...

C55FMC memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
78	UMISR register (C55FMC_UM8)	32	R/W	0000_0000h	26.5.13/939
7C	UMISR register (C55FMC_UM9)	32	R/W	0000_0000h	26.5.14/940
80	Over-Program Protection 0 register (C55FMC_OPP0)	32	R	See section	26.5.15/941
84	Over-Program Protection 1 register (C55FMC_OPP1)	32	R	See section	26.5.16/942
88	Over-Program Protection 2 register (C55FMC_OPP2)	32	R	See section	26.5.17/943
8C	Over-Program Protection 3 register (C55FMC_OPP3)	32	R	See section	26.5.18/943
90	Test Mode Disable Password Check register (C55FMC_TMD)	32	R/W	0000_0000h	26.5.19/944

26.5.1 Module Configuration Register (C55FMC_MCR)

NOTE

1. A number of Module Configuration Register (MCR) bits are locked against write by other bits. These locks are discussed in relationship to each bit in this section. Simultaneously writing bits which lock each other out is also discussed in [Functional Description](#).
2. See [Functional Description](#) for information about simultaneous MCR writes, and priority levels.

Address: 0h base + 0h offset = 0h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	RVE	RRE	AEE	EEE	0											
W	w1c	w1c	w1c	w1c												
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	
R	EER	RWE	SBC	0	PEAS	DONE	PEG	PECIE	0				PGM	PSUS	ERS	ESUS	EHV
W	w1c	w1c	w1c														
Reset	0	0	0	0	0	1	1	0	0	0	0	0	0	0	0	0	

C55FMC_MCR field descriptions

Field	Description
0 RVE	Read Voltage Error RVE provides information on previous reads monitoring the read pump voltage. If the read voltage is detected to be out of range, this bit is set to indicate that previous reads requested may have been

Table continues on the next page...

C55FMC_MCR field descriptions (continued)

Field	Description
	corrupted. Since this bit is a status flag, it must be cleared to a 0 by writing a 1 to the register location. A write of 0 has no effect. 0 Reads are occurring without voltage issues 1 A previous read may have been corrupted due to read voltage being out of range
1 RRE	Read Reference Error RRE provides information on previous reads monitoring the read reference. If the read reference is detected to be out of range, this bit is set to indicate that previous reads requested may have been corrupted. Since this bit is a status flag, it must be cleared to a 0 by writing a 1 to the register location. A write of 0 has no effect. 0 Reads are occurring without reference issues 1 A previous read may have been corrupted due to read reference being out of range
2 AEE	Address Encode Error AEE provides information on previous reads monitoring the address encode feature. On every read request to the flash, the incoming address is compared to an encoded address (row, column, and block) coming back from the memory array using the read data sense amplifier timing. If these two values do not match (zero selected, multiple selected, wrong selected), or the timing is incorrect, an address encode error will be recorded. If an address encode mismatch is detected, this bit is set to indicate that previous reads requested may have been corrupted. Since this bit is a status flag, it must be cleared to a 0 by writing a 1 to the register location. A write of 0 has no effect. 0 Reads are occurring without address encode mismatches 1 A previous read may be corrupted based on address encode mismatch
3 EEE	ECC after ECC Error EEE provides information on previous reads monitoring the ECC after ECC feature. On every read request to the flash, ECC is recalculated serially, and if there is a mismatch between the ECC calculations (taking into account corrections or detections) a late error will be reported. If an ECC after ECC mismatch is detected, this bit is set to indicate that previous reads requested may have been corrupted. Since this bit is a status flag, it must be cleared to a 0 by writing a 1 to the register location. A write of 0 has no effect. 0 Reads are occurring without ECC after ECC mismatches 1 A previous read may be corrupted based on ECC calculation errors
4–15 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
16 EER	ECC Event Error This bit provides information on previous reads. If a double bit detection occurred, the EER bit is set to a '1'. This bit must then be cleared, or a reset must occur before it returns to a 0 state. This bit may not be set by software. In the event of a single bit detection and correction, this bit is not set. If EER is not set, or remains 0, this indicates that all previous reads (from the last reset, or clearing of EER) are correct. Since this bit is an error flag, it must be cleared to a 0 by writing a 1 to the register location. A write of 0 has no effect. 0 Reads are occurring normally. 1 An ECC Error occurred during a previous read.
17 RWE	Read-While-Write Event Error This bit provides information on previous read-while-write (RWW) reads. If an RWW error occurs, this bit is set to 1. The bit must then be cleared, or a reset must occur before it returns to a 0 state. This bit may not be written to a 1. If RWE is not set, or remains 0, this indicates that all previous RWW reads (from the last

Table continues on the next page...

C55FMC_MCR field descriptions (continued)

Field	Description
	reset, or clearing of RWE) are correct. Since this bit is an error flag, it must be cleared to a 0 by writing a 1 to the register location. A write of 0 has no effect. 0 Reads are occurring normally. 1 A read-while-write error occurred during a previous read.
18 SBC	Single Bit Correction SBC provides information on previous reads, if the SBCE is set. If a single bit correction occurred, the SBC bit is set to a 1. This bit must then be cleared, or a reset must occur before this bit returns to a 0 state. If SBC is not set, or remains 0, this indicates that all previous reads (from the last reset, or clearing of SBC) did not require a correction. Since this bit is a status flag, it must be cleared to a 0 by writing a 1 to the register location. A write of 0 has no effect. 0 Reads are occurring without corrections. 1 A Single Bit Correction occurred during a previous read.
19 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
20 PEAS	Program Access Space PEAS indicates which space is valid for program or erase operations — either main array space or UTest NVM space. It should be noted that if the flash is sealed, and the Test Mode Disable Seal is programmed, erase is not allowed on the UTest NVM space. PEAS = 0 indicates the main address space is active for all program or erase operations. PEAS = 1 indicates the UTest NVM address space is active for program or erase (if applicable) operations. The value in PEAS is captured and held when the UTest NVM block is enabled with the first interlock write done for program operations (and erase operations if the flash is unsealed). The value of PEAS is retained between sampling events (in other words, subsequent first interlock writes). The value in PEAS may be changed during erase-suspended program, and reverts back to its original state once the erase-suspended program is completed. PEAS is read only. 0 UTest NVM address space is disabled for program/erase and main address space enabled. 1 UTest NVM address space is enabled for program/erase and main address space disabled.
21 DONE	State Machine Status DONE indicates whether the embedded flash memory is performing a high voltage operation. DONE is set to a 1 on termination of the embedded flash memory reset. DONE is read only. DONE is set to a 1 at the end of program and erase high voltage sequences. NOTE: This bit transitions from a 0 to 1 during reset and remains at 1 after reset. 0 Flash memory is executing a high voltage operation. 1 Flash memory is not executing a high voltage operation.
22 PEG	Program/Erase Good The PEG bit indicates the completion status of the last flash memory program or erase sequence for which high voltage operations were initiated. The value of PEG is updated automatically during the program and erase high voltage operations. Aborting a program/erase high voltage operation causes PEG to be cleared, indicating the sequence failed. PEG is set to a 1 when the embedded flash memory is reset. PEG is read only. The value of PEG is valid only when PGM = 1 or ERS = 1 and after DONE transitions from 0 to 1 due to an abort or the completion of a program/erase operation. PEG is valid until PGM/ERS makes a 1 to 0 transition or EHV makes a 0 to 1 transition. The value in PEG is not valid after a 0 to 1 transition of DONE caused by PSUS or ESUS being set to logic 1. If PGM and ERS are both 1 when DONE makes a qualifying 0 to 1 transition the value of PEG indicates the completion status of the PGM sequence. This happens in an erase-suspended program operation.

Table continues on the next page...

C55FMC_MCR field descriptions (continued)

Field	Description
	PEG will be 0 if an Erase-Suspended Program is attempted to a block that has been selected for Erase. PEG will also be 0 in the event of an error in the interlock write for program only. An error in the interlock write occurs when the ECC value provided to the flash memory does not match the ECC calculated within the flash memory. In the event of this, high voltage will not be applied to the array, and PEG will indicate a failed program. NOTE: 1. If program or erase operations are attempted on blocks that are locked, the response from embedded flash memory is PEG = 1, indicating that the operation was successful, and the contents of the block are properly protected from the program or erase operation. PEG = 1 is also true if an abort occurs during an HV request to a locked block. 2. If a program operation is attempted to a location marked as OTP, the response from the embedded flash memory is PEG = 0, indicating that the operation was not allowed, and the value interlocked was not programmed, since the desired doubleword was already programmed with a previous program operation. 0 Program or erase operation failed. 1 Program or erase operation successful.
23 PECIE	Program/Erase Complete Interrupt Enable PECIE provides a mechanism to trigger an interrupt request upon the assertion of the DONE flag due to a high voltage event (program or erase) finishing (normal, abort, suspend). If PECIE is written while not in a high voltage event, the interrupt will not immediately trigger, but will trigger after the next high voltage event is completed. 0 An interrupt request will not be generated when the DONE flag is set. 1 An interrupt request will be generated when the DONE flag is set.
24–26 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
27 PGM	Program PGM is used to set up embedded flash memory for a program operation. A 0 to 1 transition of PGM initiates a program sequence. A 1 to 0 transition of PGM ends the program sequence. PGM can be set only under one of the following conditions: • User mode read (ERS is low and UTE is low) • Erase suspend (ERS and ESUS are 1) with EHV low PGM can be cleared only when PSUS and EHV are low and DONE is high. PGM is cleared on reset. 0 Flash memory is not executing a program sequence. 1 Flash memory is executing a program sequence.
28 PSUS	Program Suspend PSUS is used to indicate the embedded flash memory is in program suspend or in the process of entering a suspend state. The embedded flash memory is in program suspend when PSUS = 1 and DONE = 1. PSUS can be set high only when PGM and EHV are high. A 0 to 1 transition of PSUS starts the sequence which sets DONE and places the embedded flash memory in program suspend. The embedded flash memory enters suspend within Tpsus of this transition. PSUS can be cleared only when DONE and EHV are high. A 1 to 0 transition of PSUS with EHV = 1 starts the sequence which clears DONE and returns the embedded flash memory to program. The embedded flash memory cannot exit program suspend and clear DONE while EHV is low. PSUS is cleared on reset. 0 Program sequence is not suspended. 1 Program sequence is suspended.

Table continues on the next page...

C55FMC_MCR field descriptions (continued)

Field	Description
29 ERS	Erase ERS is used to set up embedded flash memory for an erase operation. A 0 to 1 transition of ERS initiates an erase sequence. A 1 to 0 transition of ERS ends the erase sequence. ERS can only be set in user mode read (PGM is low and UTE is low). ERS can be cleared only when ESUS and EHV are low and DONE is high. ERS is cleared on reset. 0 Flash memory is not executing an erase sequence. 1 Flash memory is executing an erase sequence.
30 ESUS	Erase Suspend ESUS is used to indicate that the embedded flash memory is in erase suspend or in the process of entering a suspend state. The embedded flash memory is in erase suspend when ESUS = 1 and DONE = 1. ESUS can be set high only when ERS and EHV are high and PGM is low. A 0 to 1 transition of ESUS starts the sequence which sets DONE and places the flash memory in erase suspend. The embedded flash memory enters suspend within Tesus of this transition. ESUS can be cleared only when DONE and EHV are high and PGM is low. A 1 to 0 transition of ESUS with EHV = 1 starts the sequence which clears DONE and returns the embedded flash memory to erase. The embedded flash memory cannot exit erase suspend and clear DONE while EHV is low. ESUS is cleared on reset. 0 Erase sequence is not suspended. 1 Erase sequence is suspended.
31 EHV	Enable High Voltage The EHV bit enables the embedded flash memory for a high voltage program/erase operation. EHV is cleared on reset. EHV must be set after an interlock write to start a program/erase sequence. EHV may be set, initiating a program/erase, after an interlock under one of the following conditions: • Erase (ERS = 1, ESUS = 0) • Program (ERS = 0, ESUS = 0, PGM = 1, PSUS = 0) • Erase-suspended program (ERS = 1, ESUS = 1, PGM = 1, PSUS = 0) If a program operation is to be initiated while an erase is suspended, EHV must be cleared while in erase suspend before setting PGM. In normal operation, a 1 to 0 transition of EHV with DONE high, PSUS low, and ESUS low terminates the current program/erase high-voltage operation. In an erase-suspended program operation, a 1 to 0 transition of EHV with DONE high, ESUS high, PGM high, ERS high, and PSUS low terminates the program that is the current high voltage operation. When an operation is aborted, there is a 1 to 0 transition of EHV with DONE low and the suspend bit for the current program/erase sequence low. An abort causes the value of PEG to be cleared, indicating a failed program/erase; address locations being operated on by the aborted operation contain indeterminate data after an abort. EHV may not be written to a 1 after an abort is requested (EHV being cleared) and before DONE transitions high. A suspended operation cannot be aborted. EHV may be written during suspend. EHV must be high for the embedded flash memory to exit suspend. EHV may not be written after a suspend bit is set high and before DONE transitions high. EHV may not be set low after the current suspend bit is set low and before DONE transitions low. NOTE: Aborting a high voltage operation leaves FC addresses in an indeterminate data state. This may be recovered by executing an erase on the affected blocks. 0 Flash memory is not enabled to perform a high voltage operation. 1 Flash memory is enabled to perform a high voltage operation.

26.5.2 Extended Module Configuration Register (C55FMC_MCRE)

NOTE

The reset value of this register is set at the factory and varies among devices. See the chip-specific C55FMC information for details.

Address: 0h base + 8h offset = 8h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	HT	0	n256K						n64Kh			n32Kh		n16Kh		
W																
Reset	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	n64Km			n32Km			n16Km			n64Kl			n32Kl		n16Kl	
W																
Reset	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*

* Notes:

- The reset value of this register is set at the factory and varies among devices. See the chip-specific C55FMC information for details.

C55FMC_MCRE field descriptions

Field	Description
0 HT	High Temperature Enabled. HT provides configuration information of the module. If the flash is tuned for high temperature operation (165 °C) at reduced performance (slower read at 165 °C and less and slower program and erase at 150 °C or less), and limited features at this temperature (no margin reads), the HT status bit will be a 1. Default is 150 °C as maximum operating temperature, and the HT status bit will be a 0. HT is read only. 0 150 °C maximum operating temperature. NOTE: 165 °C operating temperature maximum is not supported on this device. 1 Reserved
1–2 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
3–7 n256K	Number of 256 KB blocks. The value of this field is dependent upon the size of the embedded flash memory. This field is read only. 00000 Zero 256 KB blocks. 00001 Two 256 KB blocks (512 KB total). 00010 Four 256 KB blocks (1 MB total). 00011 Six 256 KB blocks (1.5 MB total). 00100 Eight 256 KB blocks (2 MB total).

Table continues on the next page...

C55FMC_MCRE field descriptions (continued)

Field	Description
	00101 Ten 256 KB blocks (2.5 MB total). 00110 Twelve 256 KB blocks (3 MB total). 00111 Fourteen 256 KB blocks (3.5 MB total). 01000 Sixteen 256 KB blocks (4 MB total). 01001 Eighteen 256 KB blocks (4.5 MB total). 01010 Twenty 256 KB blocks (5 MB total). 01011 Twenty-two 256 KB blocks (5.5 MB total). 01100 Twenty-four 256 KB blocks (6 MB total). 01101 Twenty-six 256 KB blocks (6.5 MB total). 01110 Twenty-eight 256 KB blocks (7 MB total). 01111 Thirty 256 KB blocks (7.5 MB total). 10000 Thirty-two 256 KB blocks (8 MB total). 10001 Reserved 10010 Reserved 10011 Reserved 10100 Reserved 10101 Reserved 10110 Reserved 10111 Reserved 11000 Forty-eight 256 KB blocks (12 MB total). 11001 Reserved. 11010 Reserved. 11011 Reserved. 11100 Reserved. 11101 Reserved. 11110 Reserved. 11111 Reserved.
8–10 n64Kh	Number of 64 KB blocks in partitions 4 and 5. This field is read only. 000 Zero 64 KB blocks. 001 Two 64 KB blocks. 010 Four 64 KB blocks. 011 Six 64 KB blocks. 100 Eight 64 KB blocks. 101 Twelve 64 KB blocks.. 110 Sixteen 64 KB blocks. 111 Reserved.
11–12 n32Kh	Number of 32 KB blocks in partitions 4 and 5. This field is read only. 00 Zero 32 KB blocks. 01 Two 32 KB blocks. 10 Four 32 KB blocks. 11 Reserved.
13–15 n16Kh	Number of 16 KB blocks in partitions 4 and 5. This field is read only. 000 Zero 16 KB blocks. 001 Two 16 KB blocks.

Table continues on the next page...

C55FMC_MCRE field descriptions (continued)

Field	Description
	010 Four 16 KB blocks. 011 Six 16 KB blocks. 100 Eight 16 KB blocks. 101 Reserved. 110 Reserved. 111 Reserved.
16–18 n64Km	Number of 64 KB blocks in partitions 2 and 3. This field is read only. 000 Zero 64 KB blocks. 001 Two 64 KB blocks. 010 Four 64 KB blocks. 011 Six 64 KB blocks. 100 Eight 64 KB blocks. 101 Reserved. 110 Reserved. 111 Reserved.
19–20 n32Km	Number of 32 KB blocks in partitions 2 and 3. This field is read only. 00 Zero 32 KB blocks. 01 Two 32 KB blocks. 10 Four 32 KB blocks. 11 Reserved.
21–23 n16Km	Number of 16 KB blocks in partitions 2 and 3. This field is read only. 000 Zero 16 KB blocks. 001 Two 16 KB blocks. 010 Four 16 KB blocks. 011 Six 16 KB blocks. 100 Eight 16 KB blocks. 101 Reserved. 110 Reserved. 111 Reserved.
24–26 n64KI	Number of 64 KB blocks in partitions 0 and 1. This field is read only. 000 Zero 64 KB blocks. 001 Two 64 KB blocks. 010 Four 64 KB blocks. 011 Six 64 KB blocks. 100 Eight 64 KB blocks. 101 Reserved. 110 Reserved. 111 Reserved.
27–28 n32KI	Number of 32 KB blocks in partitions 0 and 1. This field is read only. 00 Zero 32 KB blocks. 01 Two 32 KB blocks. 10 Four 32 KB blocks. 11 Reserved.

Table continues on the next page...

C55FMC_MCRE field descriptions (continued)

Field	Description
29–31 n16KI	Number of 16 KB blocks in partitions 0 and 1. This field is read only. 000 Zero 16 KB blocks. 001 Two 16 KB blocks. 010 Four 16 KB blocks. 011 Six 16 KB blocks. 100 Reserved. 101 Reserved. 110 Reserved. 111 Reserved.

26.5.3 Lock 0 register (C55FMC_LOCK0)

The Lock 0 (LOCK0) register provides a means to protect blocks from being modified.

Address: 0h base + 10h offset = 10h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	TSLOCK	0	LOWLOCK													
W																
Reset	1	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1
Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	MIDLOCK															
W																
Reset	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

C55FMC_LOCK0 field descriptions

Field	Description
0 TSLOCK	UTest NVM Lock. This bit is used to lock the UTest NVM block from programs (erase protection not needed since UTest NVM is OTP and not erasable). A value of 1 in the TSLOCK register signifies that the UTest NVM block is locked for program. A value of 0 in the TSLOCK register signifies that the UTest NVM block is available to receive program pulses. The TSLOCK register is not writable once an interlock write is completed until DONE is set at the completion of the requested operation. Likewise, TSLOCK register is not writable if a high voltage operation is suspended.

Table continues on the next page...

C55FMC_LOCK0 field descriptions (continued)

Field	Description
	0 UTest NVM block is available to be programmed. 1 UTest NVM block is locked and cannot be programmed.
1 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
2–15 LOWLOCK	Low Block Lock A value of 1 in a bit of the lock register signifies that the corresponding block is locked for program and erase. A value of 0 in the lock register signifies that the corresponding block is available to receive program and erase pulses. The block numbering for the Low Blocks starts at LOWLOCK[0] with 16 KB block region (if available), then the 32 KB block region (if available), and lastly the 64 KB block region (if available). This continues until all blocks are accounted for. Low Blocks exist in partitions 0 and 1. The lock register is not writable once an interlock write is completed until DONE is set at the completion of the requested operation. Likewise, the lock register is not writable if a high voltage operation is suspended. The default value of the LOCK bits is locked. In the event that blocks are not present (due to configuration or total memory size), the LOCK bits default to be locked, and are not writable. The reset value is always 1, and register writes have no effect. 0 Block is available for program and erase operations. 1 Block is locked and unavailable for program and erase operations
16–31 MIDLOCK	Mid Block Lock A value of 1 in a bit of the lock register signifies that the corresponding block is locked for program and erase. A value of 0 in the lock register signifies that the corresponding block is available to receive program and erase pulses. The block numbering for the Mid Blocks starts at MIDLOCK[0] with 16 KB block region (if available), then the 32 KB block region (if available), and lastly the 64 KB block region (if available). This continues until all blocks are accounted for. Mid Blocks exist in partitions 2 and 3. The lock register is not writable once an interlock write is completed until DONE is set at the completion of the requested operation. Likewise, the lock register is not writable if a high-voltage operation is suspended. The default value of the LOCK bits is locked. In the event that blocks are not present (due to configuration or total memory size), the LOCK bits default to be locked, and are not writable. The reset value is always 1, and register writes have no effect. 0 Block is available for program and erase operations. 1 Block is locked and unavailable for program and erase operations

26.5.4 Lock 1 register (C55FMC_LOCK1)

The Lock 1 (LOCK1) register provides a means to protect blocks from being modified.

Address: 0h base + 14h offset = 14h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0																HIGHLOCK															
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	

C55FMC_LOCK1 field descriptions

Field	Description
0–15 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
16–31 HIGHLOCK	High Block Lock A value of 1 in a bit of the lock register signifies that the corresponding block is locked for program and erase. A value of 0 in the lock register signifies that the corresponding block is available to receive program and erase pulses. The block numbering for the High Blocks starts at HIGHLOCK[0] with 16 KB block region (if available), then the 32 KB block region (if available), and lastly the 64 KB block region (if available). This continues until all blocks are accounted. HIGH Blocks exist in partitions 4 and 5. The lock register is not writable once an interlock write is completed until DONE is set at the completion of the requested operation. Likewise, the lock register is not writable if a high-voltage operation is suspended. The default value of the LOCK bits is locked. In the event that blocks are not present (due to configuration or total memory size), the LOCK bits default to be locked, and are not writable. The reset value is always 1, and register writes have no effect. 0 Block is available for program and erase operations. 1 Block is locked and unavailable for program and erase operations

26.5.5 Lock 2 register (C55FMC_LOCK2)

The Lock 2 (LOCK2) register provides a means to protect blocks from being modified.

Address: 0h base + 18h offset = 18h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R																																
W																																
	A256KLOCK																															
Reset	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	

C55FMC_LOCK2 field descriptions

Field	Description
0–31 A256KLOCK	256 KB Block Lock 256KLOCK has the same characteristics as LOWLOCK. Please see that description for more information. The block numbering for the 256 KB blocks starts with 256KLOCK[0] and continues until all blocks are accounted. A value of 1 in a bit of the lock register signifies that the corresponding block is locked for program and erase. A value of 0 in the lock register signifies that the corresponding block is available to receive program and erase pulses. The block numbering for the 256 KB blocks starts with 256KLOCK[0] and continues until all blocks are accounted. The lock register is not writable once an interlock write is completed until DONE is set at the completion of the requested operation. Likewise, the lock register is not writable if a high-voltage operation is suspended. The default value of the LOCK bits is locked. In the event that blocks are not present (due to configuration or total memory size), the LOCK bits default to be locked, and are not writable. The reset value is always 1, and register writes have no effect. 0 Block is available for program and erase operations. 1 Block is locked and unavailable for program and erase operations

26.5.6 Lock 3 register (C55FMC_LOCK3)

The Lock 3 (LOCK3) register provides a means to protect blocks from being modified.

Address: 0h base + 1Ch offset = 1Ch

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0																A256KLOCK															
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	

C55FMC_LOCK3 field descriptions

Field	Description
0–15 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
16–31 A256KLOCK	256 KB Block Lock 256KLOCK has the same characteristics as LOWLOCK. Please see that description for more information. The block numbering for the 256 KB blocks starts with 256KLOCK[0] and continues until all blocks are accounted. A value of 1 in a bit of the lock register signifies that the corresponding block is locked for program and erase. A value of 0 in the lock register signifies that the corresponding block is available to receive program and erase pulses. The block numbering for the 256 KB blocks starts with 256KLOCK[0] and continues until all blocks are accounted.

Table continues on the next page...

C55FMC_LOCK3 field descriptions (continued)

Field	Description
	The lock register is not writable once an interlock write is completed until DONE is set at the completion of the requested operation. Likewise, the lock register is not writable if a high-voltage operation is suspended. The default value of the LOCK bits is locked. In the event that blocks are not present (due to configuration or total memory size), the LOCK bits default to be locked, and are not writable. The reset value is always 1, and register writes have no effect. 0 Block is available for program and erase operations. 1 Block is locked and unavailable for program and erase operations

26.5.7 Select 0 register (C55FMC_SEL0)

The Select 0 (SEL0) register provides a means to select blocks to be operated on during erase.

Address: 0h base + 38h offset = 38h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0		LOWSEL														MIDSEL															
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	

C55FMC_SEL0 field descriptions

Field	Description
0–1 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
2–15 LOWSEL	LOW Block Select. A value of 1 in the select register signifies that the block is selected for erase. A value of 0 in the select register signifies that the block is not selected. The reset value for the select registers is 0, or unselected. The block numbering for the Low Blocks starts at LOWSEL[0] with 16 KB block region (if available), then the 32 KB block region (if available), and lastly the 64 KB block region (if available). This continues until all blocks are accounted. Low Blocks exist in partitions 0 and 1. The blocks must be selected (or unselected) before doing an erase interlock write as part of the erase sequence. The select register is not writable once an interlock write is completed until DONE is set at the completion of the requested operation, or if a high-voltage operation is suspended. LOWSEL is also not writeable during UTest operations, when AIE is high. In the event that blocks are not present (due to configuration or total memory size), the corresponding select bits default to unselected, and are not writable. The reset value is always 0, and register writes have no effect. 0 The block is not selected 1 The block is selected for erase

Table continues on the next page...

C55FMC_SEL0 field descriptions (continued)

Field	Description
16–31 MIDSEL	Mid Block Select. A value of 1 in the select register signifies that the block is selected for erase. A value of 0 in the select register signifies that the block is not selected. The reset value for the select registers is 0, or unselected. The block numbering for the Mid Blocks starts at MIDLSEL[0] with 16 KB block region (if available), then the 32 KB block region (if available), and lastly the 64 KB block region (if available). This continues until all blocks are accounted. Mid Blocks exist in partitions 2 and 3. The blocks must be selected (or unselected) before doing an erase interlock write as part of the erase sequence. The select register is not writable once an interlock write is completed until DONE is set at the completion of the requested operation, or if a high-voltage operation is suspended. MIDSEL is also not writable during UTest operations, when AIE is high. In the event that blocks are not present (due to configuration or total memory size), the corresponding select bits default to unselected, and are not writable. The reset value is always 0, and register writes have no effect. 0 The block is not selected 1 The block is selected for erase

26.5.8 Select 1 register (C55FMC_SEL1)

The Select 1 (SEL1) register provides a means to select blocks to be operated on during erase.

Address: 0h base + 3Ch offset = 3Ch

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0																HIGHSEL															
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	

C55FMC_SEL1 field descriptions

Field	Description
0–15 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
16–31 HIGHSEL	High Block Select. A value of 1 in the select register signifies that the block is selected for erase. A value of 0 in the select register signifies that the block is not selected. The reset value for the select registers is 0, or unselected. The block numbering for the High Blocks starts at HIGHSEL[0] with 16 KB block region (if available), then the 32 KB block region (if available), and lastly the 64 KB block region (if available). This continues until all blocks are accounted. HIGH Blocks exist in partitions 4 and 5.

Table continues on the next page...

C55FMC_SEL1 field descriptions (continued)

Field	Description
	The blocks must be selected (or unselected) before doing an erase interlock write as part of the erase sequence. The select register is not writable once an interlock write is completed until DONE is set at the completion of the requested operation, or if a high-voltage operation is suspended. HIGHSEL is also not writeable during UTest operations, when AIE is high. In the event that blocks are not present (due to configuration or total memory size), the corresponding select bits default to unselected, and are not writable. The reset value is always 0, and register writes have no effect. 0 The block is not selected for erase 1 The block is selected for erase

26.5.9 Select 2 register (C55FMC_SEL2)

The Select 2 (SEL2) register provides a means to select blocks to be operated on during erase.

Address: 0h base + 40h offset = 40h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R																																
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

C55FMC_SEL2 field descriptions

Field	Description
0–31 A256KSEL	256 KB Block Select. A value of 1 in the select register signifies that the block is selected for erase. A value of 0 in the select register signifies that the block is not selected. The reset value for the select registers is 0, or unselected. The block numbering for the 256 KB blocks starts with 256KSEL[0] and continues until all blocks are accounted. The blocks must be selected (or unselected) before doing an erase interlock write as part of the erase sequence. The select register is not writable once an interlock write is completed until DONE is set at the completion of the requested operation, or if a high-voltage operation is suspended. 256KSEL is also not writeable during UTest operations, when AIE is high. In the event that blocks are not present (due to configuration or total memory size), the corresponding select bits default to unselected, and are not writable. The reset value is always 0, and register writes have no effect. 0 The block is not selected 1 The block is selected for erase

26.5.10 Select 3 register (C55FMC_SEL3)

The Select 3 (SEL3) register provides a means to select blocks to be operated on during erase.

Address: 0h base + 44h offset = 44h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	Reserved																A256KSEL															
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	

C55FMC_SEL3 field descriptions

Field	Description
0–15 Reserved	This field is reserved. Reserved, reset to 0.
16–31 A256KSEL	256 KB Block Select. A value of 1 in the select register signifies that the block is selected for erase. A value of 0 in the select register signifies that the block is not selected. The reset value for the select registers is 0, or unselected. The block numbering for the 256 KB blocks starts with 256KSEL[0] and continues until all blocks are accounted. The blocks must be selected (or unselected) before doing an erase interlock write as part of the erase sequence. The select register is not writable once an interlock write is completed until DONE is set at the completion of the requested operation, or if a high-voltage operation is suspended. 256KSEL is also not writeable during UTest operations, when AIE is high. In the event that blocks are not present (due to configuration or total memory size), the corresponding select bits default to unselected, and are not writable. The reset value is always 0, and register writes have no effect. 1 The block is selected for erase. 0 The block is not selected.

26.5.11 Address register (C55FMC_ADR)

The Address register (ADR) provides the first failing address in the event of a failure (ECC or PGM/Erase state machine), as well as single bit correction information.

Address: 0h base + 50h offset = 50h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	SAD	aH	aM	aL	a256k	a64k	a32k	a16k	ADDR							
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	ADDR												0			
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

C55FMC_ADR field descriptions

Field	Description
0 SAD	UTest NVM Address. The SAD bit qualifies the address captured during an ECC Event Error, Single Bit Correction, or State Machine operation. SAD has the same characteristics as ADDR related to capturing of addresses. The SAD field is not writable. 0 Address captured is from main array Space. 1 Address captured is from UTest NVM array space.
1 aH	Address High region. The aH bit qualifies the address field (ADDR) to the region of a16K, a32K and a64k. If aH is set, then the ADDR field maps to that region. See ADDR description for more information. 0 Address captured or to be accessed is not from high address region. 1 Address captured or to be accessed is from high address region.
2 aM	Address Mid region. The aM bit qualifies the address field (ADDR) to the region of a16K, a32K and a64k. If aM is set, then the ADDR field maps to that region. See ADDR description for more information.

Table continues on the next page...

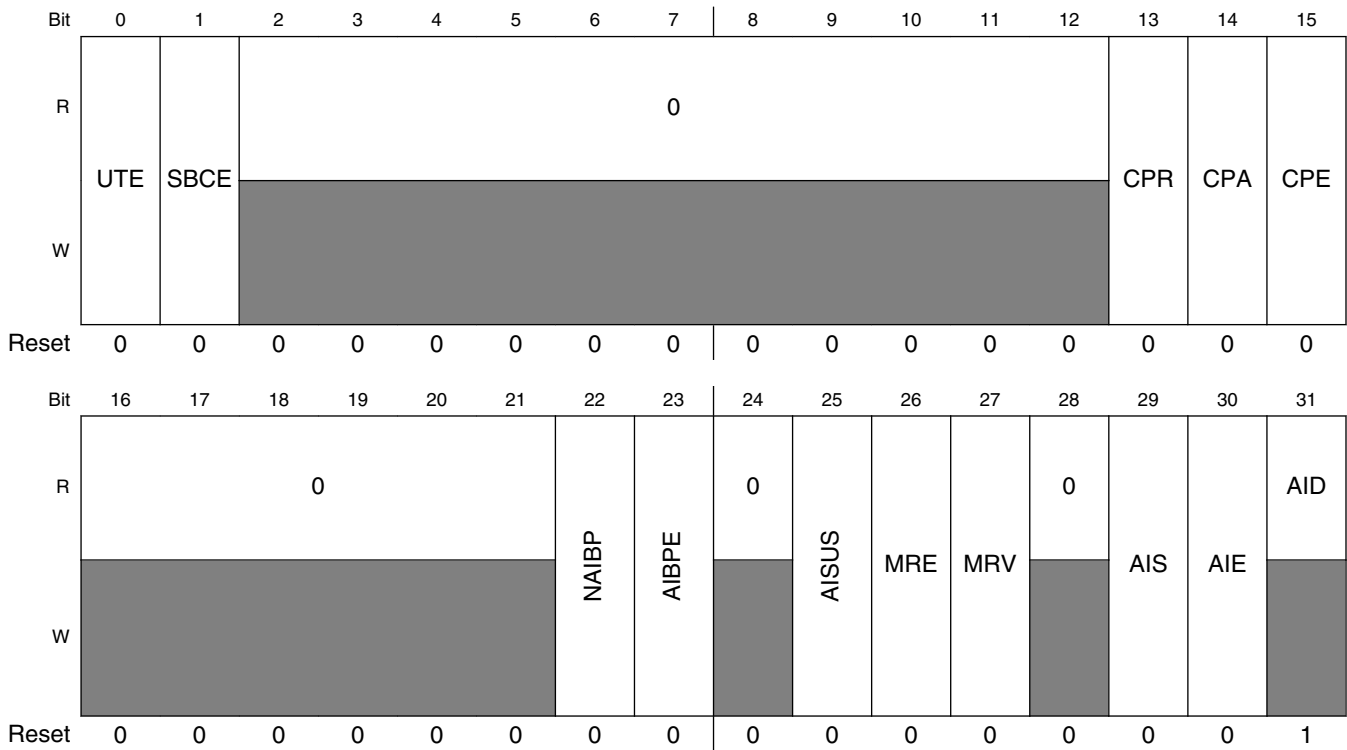
C55FMC_ADR field descriptions (continued)

Field	Description
	0 Address captured or to be accessed is not from mid address region. 1 Address captured or to be accessed is from mid address region.
3 aL	Address Low region. The aL bit qualifies the address field (ADDR) to the region of a16K, a32K and a64k. If aL is set, then the ADDR field maps to that region. See ADDR description for more information. 0 Address captured or to be accessed is not from low address region. 1 Address captured or to be accessed is from low address region.
4 a256k	Address 256 KB region. The a256k bit qualifies the address field (ADDR) to the region. If a256k is set, then the ADDR field maps to that region. See ADDR description for more information. The a256k register is writable. 0 Address captured or to be accessed is not from 256 KB region. 1 Address captured or to be accessed is from 256 KB region.
5 a64k	Address 64 KB region. The a64k bit qualifies the address field (ADDR) to the region. If a64k is set, then the ADDR field maps to that region. See ADDR description for more information. 0 Address captured or to be accessed is not from 64 KB region. 1 Address captured or to be accessed is from 64 KB region.
6 a32k	Address 32 KB region. The a32k bit qualifies the address field (ADDR) to the region. If a32k is set, then the ADDR field maps to that region. See ADDR description for more information. 0 Address captured or to be accessed is not from 32 KB region. 1 Address captured or to be accessed is from 32 KB region.
7 a16k	Address 16 KB region. The a16k bit qualifies the address field (ADDR) to the region. If a16k is set, then the ADDR field maps to that region. See ADDR description for more information. 0 Address captured or to be accessed is not from 16 KB region. 1 Address captured or to be accessed is from 16 KB region.
8–28 ADDR	Address. The ADR provides the first failing address in the event of ECC event error (EER set), Single Bit Correction (SBC set), as well as providing the address of a failure that may have occurred in a state machine operation (PEG cleared). ECC event errors take priority over Single Bit Corrections, which take priority over state machine errors. This is especially valuable in the event of an RWW operation, where the read senses an ECC error or Single Bit correction, and the state machine fails simultaneously. The failing address for ECC event error is held until EER is cleared. The failing address for Single Bit Correction is held until an ECC event error, or until SBC is cleared. State machine address is held until an ECC event error or Single Bit Correction event occurs, or until the next state machine event (PEG cleared) occurs. This address is always a doubleword address that selects 64 bits. ADDR[23:3] is an offset from a base address of 0x0 for each block size region. The aH, aM, aL, a16k, a32k, a64k, and a256k qualify the block size region the ADDR field.
29–31 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

26.5.12 UTest 0 register (C55FMC_UT0)

The User Test 0 (UT0) register provides a means to control UTest. UTest mode gives the ability to perform test features on the flash memory. This register is only writable when the flash memory is put into UTest mode by writing a passcode.

Address: 0h base + 54h offset = 54h



C55FMC_UT0 field descriptions

Field	Description
0 UTE	U-Test Enable. This status bit gives indication when U-Test is enabled. All bits in UT0, UM0, UM1, UM2, UM3, UM4, UM5, UM6, UM7, UM8 and UM9 are locked when this bit is 0. This bit is not writeable to a 1, but may be cleared. The reset value is 0. The method to set this bit is to provide a password, and if the password matches, the UTE bit is set to reflect the status of enabled, and is enabled until it is cleared by a register write. The UTE password will only be accepted if PGM = 0 and ERS = 0 (program and erase are not being requested). UTE can only be cleared if AID = 1, MRE and AIE = 0. While clearing UTE, writes to set AIE or MRE will be ignored. For UTE, the password 0xF9F9_9999 must be written to the UT0 register. 0 All bits in the UT0, UM0, UM1, UM2, UM3, UM4, UM5, UM6, UM7, UM8 and UM9 registers are locked 1 Bits in the UT0, UM0, UM1, UM2, UM3, UM4, UM5, UM6, UM7, UM8 and UM9 registers are unlocked
1 SBCE	Single Bit Correction Enable. SBCE enables Single Bit Correction results to be observed in SBC. ECC corrections that occur when SBCE is cleared will not be logged.

Table continues on the next page...

C55FMC_UT0 field descriptions (continued)

Field	Description
	0 Single Bit Corrections observation is disabled. 1 Single Bit Correction observation is enabled.
2–12 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
13 CPR	Customer Programmable Read Voltage and Reference Detection. This bit is used to control the error generation for the Customer Programmable Detection area in the UTest block. If this bit is set, and the Customer Programmable Detection location in UTest is read, a Read Voltage and Reference Error detection will be noted. This bit may not be set while CPE and CPA are set. If CPE or CPA are high, this bit will be locked. If attempts are made to write CPR, CPA or CPE simultaneously, none of them will be written. 0 Customer Programmable Read Voltage and Reference Detection Error is not enabled 1 Customer Programmable Read Voltage and Reference Detection Error is enabled
14 CPA	Customer Programmable Address Encode Detection. This bit is used to control the error generation for the Customer Programmable Detection area in the UTest block. If this bit is set, and the Customer Programmable Detection location in UTest is read, a Address Encode Error detection will be noted. This bit may not be set while CPE and CPR are set. If CPE or CPR are high, this bit will be locked. If attempts are made to write CPR, CPA or CPE simultaneously, none of them will be written. 0 Customer Programmable Address Encode Detection Error is not enabled 1 Customer Programmable Address Encode Detection Error is enabled
15 CPE	Customer Programmable EDC after ECC Detection. This bit is used to control the error generation for the Customer Programmable Detection area in the UTest block. If this bit is set, and the Customer Programmable Detection location in UTest is read, a EDC after ECC Error detection will be noted. This bit may not be set while CPR and CPA are set. If CPR or CPA are high, this bit will be locked. If attempts are made o write CPR, CPA or CPE simultaneously, none of them will be written. 0 Customer Programmable EDC after ECC Detection Error is not enabled 1 Customer Programmable EDC after ECC Detection Error is enabled
16–21 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
22 NAIBP	Next Array Integrity Break Point. If AIBPE is set, NAIBP will be set once a single bit correction (if enabled) or double bit detection is noted during the Array Integrity test. NAIBP is not writable to 1, but may be cleared to 0. Clearing NAIBP to 0, will enable the Array Integrity operation to be re-started after a breakpoint is encountered. If the Array Integrity operation completes without encountering another correction or detection, AID will be set with NAIBP remaining 0. 1 Array Integrity state machine is at a break point. 0 Array Integrity state machine is not currently at a break point.
23 AIBPE	Array Integrity Break Point Enable. If you want to enable breakpoints during an array integrity test, AIBPE may be set. See NAIBP description for more information about array integrity breakpoints. 1 Array Integrity breakpoints are enabled during Array Integrity Checks. 0 Array Integrity breakpoints are not enabled.
24 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
25 AISUS	Array Integrity Suspend. AISUS enables a suspend of an Array Integrity Operation. Array Integrity may be suspend by Setting AISUS, and resumed by clearing AISUS. AISUS is writeable only when AID is low. AISUS is clearable only when AID is high. Attempting to write AISUS and AIE on the same clock cycle will result in only AIE getting written.

Table continues on the next page...

C55FMC_UT0 field descriptions (continued)

Field	Description
	1 Array integrity sequence is suspended. 0 Array integrity sequence not suspended.
26 MRE	Margin Read Enable. MRE combined with MRV enables user margin reads to be done. Normal user reads are not affected by MRE, although user reads while the margin read operation is ongoing are not supported. MRE is not writable if AID is low, or if AISUS is high, or if NAIBP is high. 1 Margin reads are enabled. 0 Margin reads are not enabled.
27 MRV	Margin Read Value. MRV selects the margin level that is being checked. Margin can be checked to an erased level (MRV=1) or to a programmed level (MRV=0). In order for this value to be valid, MRE must also be set. MRV is not writable if AID is low, or if AISUS is high, or if NAIBP is high. 1 One's margin reads are requested. 0 Zero's margin reads are requested.
28 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
29 AIS	Array Integrity Sequence. AIS determines the address sequence to be used during array integrity checks. The default sequence (AIS = 0) is meant to replicate sequences that normal user code follows, and thoroughly checks the read propagation paths. This sequence is proprietary. The alternative sequence (AIS = 1) is logically sequential. It should be noted that the time to run a sequential sequence is shorter than the time to run the proprietary sequence. AIS is not writeable if AIE is high. 1 Array integrity sequence is sequential. 0 Array integrity sequence is proprietary sequence.
30 AIE	Array Integrity Enable. AIE set to one starts the array integrity check done on all selected blocks. The address sequence is determined by AIS, and the MISR (UM0 through UM9) can be checked after the operation is complete, to determine if a correct signature has been obtained. Once an array integrity operation is requested (AIE=1), it may be terminated by clearing AIE if the operation has finished (AID = 1) or aborted by clearing AIE if the operation is ongoing (AID = 0). AIE is not simultaneously writable to a 1 as UTE is being cleared to a 0. 0 Array integrity checks are not enabled. 1 Array integrity checks are enabled.
31 AID	Array Integrity Done. AID is cleared upon an array integrity check being enabled (to signify the operation is ongoing). Once completed, AID is set to indicate that the array integrity check is complete. At this time the MISR (UMR registers) can be checked. AID may also assert if breakpoints are enabled (AIBPE is set), an abort is requested or a suspend is requested. AID cannot be written, and is status only. 0 Array integrity check is ongoing. 1 Array integrity check is done.

26.5.13 UMISR register (C55FMC_UMn)

The Multiple Input Signature registers provide a means to evaluate array integrity.

Address: 0h base + 58h offset + (4d × i), where i=0d to 8d

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	MISR																															
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

C55FMC_UMn field descriptions

Field	Description
0–31 MISR	MISR[31:0]. The MISR registers accumulate a signature from an array integrity event. The MISR captures all data fields, as well as ECC fields, and the ECC error signal. Data (read and ECC) captured in the MISR is after the ECC logic, and represents corrected data (if required). The MISR can be seeded to any value by writing the MISR registers. NOTE: Writing the MISR register during an array integrity operation (including suspend or breakpoint) although possible, is not recommended. A write of the MISR registers at this point may alter the final signature in an unpredictable way. The MISR register provides a means to calculate a MISR during array integrity operations. The MISR can be represented by the following polynomial: $x^{289} + x^7 + x^6 + x^5 + x^4 + x^2 + 1$ The MISR is calculated by taking the previous MISR value and then "exclusive ORing" the new data. In addition the most significant bit (in this case it is MISR[288]), is then "exclusive ORed" into input of MISR[7], MISR[6], MISR[5], MISR[4], MISR[2] and MISR[0]. The result of the "exclusive OR" is shifted left on each read. The MISR register is used in array integrity operations. If during address sequencing, reads extend into an invalid address location (in other words, greater than the maximum address for a given array size) then reads are still executed to the array, but the results from the array read are not deterministic. In this instance, the MISR registers is not recalculated, and the previous value is retained.

26.5.14 UMISR register (C55FMC_UM9)

The Multiple Input Signature registers provide a means to evaluate array integrity.

Address: 0h base + 7Ch offset = 7Ch

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0															MISR
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

C55FMC_UM9 field descriptions

Field	Description
0–30 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
31 MISR	MISR[288]. The MISR registers accumulate a signature from an array integrity event. The MISR captures all data fields, as well as ECC fields, and the ECC error signal. Data (read and ECC) captured in the MISR is after the ECC logic, and represents corrected data (if required). The MISR can be seeded to any value by writing the MISR registers. NOTE: Writing the MISR register during an array integrity operation (including suspend or breakpoint) although possible, is not recommended. A write of the MISR registers at this point may alter the final signature in an unpredictable way. The MISR register provides a means to calculate a MISR during array integrity operations. The MISR can be represented by the following polynomial: $x^{289} + x^7 + x^6 + x^5 + x^4 + x^2 + 1$ The MISR is calculated by taking the previous MISR value and then "exclusive ORing" the new data. In addition the most significant bit (in this case it is MISR[288]), is then "exclusive ORed" into input of MISR[7], MISR[6], MISR[5], MISR[4], MISR[2] and MISR[0]. The result of the "exclusive OR" is shifted left on each read. The MISR register is used in array integrity operations. If during address sequencing, reads extend into an invalid address location (in other words, greater than the maximum address for a given array size) then reads are still executed to the array, but the results from the array read are not deterministic. In this instance, the MISR registers is not recalculated, and the previous value is retained

26.5.15 Over-Program Protection 0 register (C55FMC_OPP0)

The Over-Program Protection 0 (OPP0) register provides a means to protect blocks from being over-programmed. This register shows the over-program protection status.

NOTE

The reset value of this register is set at the factory and varies among devices. See the chip-specific C55FMC information for details.

Address: 0h base + 80h offset = 80h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	LOWOPP															MIDOPP															
W																																
Reset	0	0	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	

* Notes:

- MIDOPP field: The reset value of the MIDOPP field is set at the factory and varies among devices. See the chip-specific C55FMC information for details.
- LOWOPP field: The reset value of the LOWOPP field is set at the factory and varies among devices. See the chip-specific C55FMC information for details.

C55FMC_OPP0 field descriptions

Field	Description
0–1 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
2–15 LOWOPP	Low Block Over-Program Protection[13:0]. A value of 1 in a bit of the over-program protection register signifies that the corresponding block is protected from over-programming. A value of 0 in the OPP register signifies that the corresponding block is available to be over-programmed. The block numbering for the Low Blocks starts at LOWOPP[0] with 16 KB block region (if available), then the 32 KB block region (if available), and lastly the 64 KB block region (if available). This continues until all blocks are accounted for. The LOWOPP register is not writable, and is status only. The default value of the LOWOPP bits is not OPP. In the event that blocks are not present (due to configuration or total memory size), the OPP bits default not OPP, and will remain 0. 1 The corresponding block is protected from over-programming. 0 The corresponding block is available to be over-programmed.
16–31 MIDOPP	Mid Block Over-Program Protection[15:0]. A value of 1 in a bit of the over-program protection register signifies that the corresponding block is protected from over-programming. A value of 0 in the OPP register signifies that the corresponding block is available to be over-programmed. The block numbering for the Mid Blocks starts at MIDOPP[0] with 16 KB block region (if available), then the 32 KB block region (if available), and lastly the 64 KB block region (if available). This continues until all blocks are accounted for.

Table continues on the next page...

C55FMC_OPP0 field descriptions (continued)

Field	Description
	The MIDOPP register is not writable, and is status only. The default value of the MIDOPP bits is not OPP. In the event that blocks are not present (due to configuration or total memory size), the OPP bits default not OPP, and will remain 0. 1 The corresponding block is protected from over-programming. 0 The corresponding block is available to be over-programmed.

26.5.16 Over-Program Protection 1 register (C55FMC_OPP1)

The Over-Program Protection 1 (OPP1) register provides a means to protect blocks from being over programmed. This register shows the over-program protection status.

NOTE

The reset value of this register is set at the factory and varies among devices. See the chip-specific C55FMC information for details.

Address: 0h base + 84h offset = 84h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0																HIGHOPP															
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*

* Notes:

- HIGHOPP field: The reset value of the HIGHOPP field is set at the factory and varies among devices. See the chip-specific C55FMC information for details.

C55FMC_OPP1 field descriptions

Field	Description
0–15 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
16–31 HIGHOPP	High Block Over-Program Protection[15:0]. A value of 1 in a bit of the over-program protection register signifies that the corresponding block is protected from over-programming. A value of 0 in the OPP register signifies that the corresponding block is available to be over-programmed. The block numbering for the High Blocks starts at HIGHOPP[0] with 16 KB block region (if available), then the 32 KB block region (if available), and lastly the 64 KB block region (if available). This continues until all blocks are accounted. The HIGHOPP register is not writable, and is status only. The default value of the HIGHOPP bits is not OPP. In the event that blocks are not present (due to configuration or total memory size), the OPP bits default not OPP, and will remain 0. 1 The corresponding block is protected from over-programming. 0 The corresponding block is available to be over-programmed.

26.5.17 Over-Program Protection 2 register (C55FMC_OPP2)

The Over-Program Protection 2 (OPP2) register provides a means to protect blocks from being over programmed. This register shows the over-program protection status.

NOTE

The reset value of this register is set at the factory and varies among devices. See the chip-specific C55FMC information for details.

Address: 0h base + 88h offset = 88h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	A256KOPP																															
W																																
Reset	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*

* Notes:

- A256KOPP field: The reset value of the A256KOPP field is set at the factory and varies among devices. See the chip-specific C55FMC information for details.

C55FMC_OPP2 field descriptions

Field	Description
0–31 A256KOPP	256K Block Over-Program Protection[31:0]. A value of 1 in a bit of the over-program protection register signifies that the corresponding block is protected from over-programming. A value of 0 in the OPP register signifies that the corresponding block is available to be over-programmed. The block numbering for the 256 KB Blocks starts at A256KOPP[0] and continues until all blocks are accounted. The 256KOPP register is not writable, and is status only. The default value of the A256KOPP bits is not OPP. In the event that blocks are not present (due to configuration or total memory size), the OPP bits default not OPP, and will remain 0. 1 The corresponding block is protected from over-programming. 0 The corresponding block is available to be over-programmed.

26.5.18 Over-Program Protection 3 register (C55FMC_OPP3)

The Over-Program Protection 3 (OPP3) register provides a means to protect blocks from being over programmed. This register shows the over-program protection status.

NOTE

The reset value of this register is set at the factory and varies among devices. See the chip-specific C55FMC information for details.

Memory map and register definition

Address: 0h base + 8Ch offset = 8Ch

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0																A256KOPP															
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*

* Notes:

- A256KOPP field: The reset value of the A256KOPP field is set at the factory and varies among devices. See the chip-specific C55FMC information for details.

C55FMC_OPP3 field descriptions

Field	Description
0–15 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
16–31 A256KOPP	256K Block Over-Program Protection[47:32]. A value of 1 in a bit of the over-program protection register signifies that the corresponding block is protected from over-programming. A value of 0 in the OPP register signifies that the corresponding block is available to be over-programmed. The block numbering for the 256 KB Blocks starts at 256KOPP[0] and continues until all blocks are accounted. The 256KOPP field is not writable, and is status only. The default value of the A256KOPP bits is not OPP. In the event that blocks are not present (due to configuration or total memory size), the OPP bits default not OPP, and will remain 0. 1 The corresponding block is protected from over-programming. 0 The corresponding block is available to be over-programmed.

26.5.19 Test Mode Disable Password Check register (C55FMC_TMD)

The Test Mode Disable Password Check (TMD) register provides a means to provide a challenge password to disable the Test Mode Disable Seal block select.

Writes to the register have no effect except for a password challenge. Reads to this register always return 0.

Address: 0h base + 90h offset = 90h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R																																
W																	PWD															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	

C55FMC_TMD field descriptions

Field	Description
0–31 PWD	Password challenge

26.6 Functional Description

The embedded flash memory consists of address spaces in four groupings:

- Low address space
- Mid address space
- High address space
- 256 KB address space

The main address space is divided into partitions. Each address space mentioned above consists of a partition pair. Partitions are used to determine locations for valid read-while-write (RWW) operations. While the embedded flash memory is performing a write (program or erase) to a given partition, it can simultaneously perform a read from any other partition. For program operations, only the address specified by an interlock write determines the partition being written (block locking and block select registers do not determine the RWW partitions being written). For erase operations, only blocks that are selected and unlocked determine the RWW partitions being written.

The main address space is also divided into blocks to implement independent erase and program protection. The UTest NVM block also exists as a block, and has independent program protection. The UTest NVM block is included to support systems that require nonvolatile memory (NVM) for security or to store system initialization information.

A number of MCR bits are protected against write when another bit, or set of bits, is in a specific state. These write locks are covered on a bit-by-bit basis in [Module Configuration Register \(C55FMC_MCR\)](#). The write locks do not consider the effects of trying to write two or more bits simultaneously. The embedded flash memory does not allow bits to be written simultaneously which put the device into an illegal state. This is implemented through a priority mechanism among the bits. The bit changing priorities are detailed in the following table.

Table 26-32. MCR bit set/clear priority levels

Priority level	MCR Bit(s)
1	ERS
2	PGM
3	EHV
4	ESUS, PSUS

If two or more MCR bits are written simultaneously only the bit with the lowest priority number is accepted for modification. Setting two bits with the same priority number is prevented by existing write locks or do not put the flash memory in an illegal state.

For example, setting ERS and PGM simultaneously results in only ERS being set. Attempting to clear EHV while setting PSUS results in EHV being cleared, while PSUS is unaffected.

Each read of the embedded flash memory retrieves a page, or eight consecutive words (256 bits), of information. The address for each word retrieved within a page differs from the other addresses in the page only by address bits [4:2]. The flash memory page read architecture easily supports both cache and burst mode at the BIU level for high-speed read applications.

The embedded flash memory supports fault tolerance through error correction code (ECC) and error detection. ECC implemented within the embedded flash memory corrects single-bit failures and detects double-bit failures.

A software mechanism is provided to independently lock and unlock each block.

Program and erase of the embedded flash memory requires multiple system clock cycles to complete. The program and erase sequence may be suspended or aborted.

The embedded flash memory may reside in various modes. The modes that are available include:

- User mode
- UTest mode

Each of these modes is discussed in detail in the following sections.

26.6.1 User Mode

In user mode the embedded flash memory may be read and written (register writes and interlock writes), programmed or erased. The following sub-sections define all actions that may be performed in user mode.

26.6.1.1 Read and write

The default state of the embedded flash memory is read. The main and UTest NVM address space can be read only in the read state. The MCR is always available for read. The embedded flash memory enters the read state on reset. The embedded flash memory is in the read state under four sets of conditions:

- The read state is active when the embedded flash memory is enabled (User Mode Read).
- The read state is active when MCR[PGM] or MCR[ERS] are high and high-voltage operation is ongoing (read-while-write).

NOTE

Reads done to the partition(s) being operated on (either erased or programmed) result in an error and the MCR[RWE] bit is set.

- The read state is active when MCR[PGM] and MCR[PSUS] are high (program suspend).
- The read state is active when MCR[ERS] and MCR[ESUS] are high and MCR[PGM] is low (erase suspend).

In embedded flash memory, flash core reads return 256 bits (1 page). Register reads return 32 bits of data.

NOTE

Flash core reads are done through the BIU. In many cases the BIU does read page buffering to allow sequential reads to be done with higher performance. This could provide a data coherency issue that must be handled with software. Data coherency may be an issue after a program or erase operation, as well as UTest NVM block operations.

In user mode, registers may be written. Array may be written to do interlock writes.

Register reads to unmapped register address space return all zeroes.

Register writes to unmapped register address space have no effect.

Interlock writes that are attempted during a high-voltage operation (MCR[EHV] = 1 or MCR[DONE] = 0) will result in the interlock write being ignored, and address and data will not be updated.

NOTE

Care must be taken when doing 32-bit interlock writes. Performing two 32-bit interlock writes to the same word location is not supported. In addition, mixing 32-bit interlock writes with 64-bit interlock writes to the same double-word location is not supported. If these combinations are done, and a program operation is executed, the result will be MCR[PEG] = 0. See the Module Configuration Register (MCR) section for more information.

26.6.1.2 Program

A flash memory program sequence operates on any page within the flash core. Within a page, up to eight words may be altered in a single program operation. Also, up to four pages can be altered in a single program operation. Whenever the array is programmed, the ECC bits also get programmed. ECC is handled on a 64-bit boundary. Thus, if only one word in any given 64-bit ECC segment is programmed, the adjoining word (in that segment) should not be programmed, since ECC calculation has already completed for that 64-bit segment. Attempts to program the adjoining word will probably result in an operation failure. It is recommended that all programming operations be from 64 bits to 1024 bits, and be 64-bit aligned. The programming operation should completely fill selected ECC segments within the page. Only one program is allowed per 64-bit ECC segment between erases.

Warning

In rare cases over-programming of a 64-bit ECC segment may be done (EEPROM emulation).

Programming changes the value stored in an array bit from logic 1 to logic 0 only. Programming cannot change a stored logic 0 to a logic 1.

NOTE

If a logic 0 is attempted to be over-programmed by a logic 1, the resulting operation will fail ($MCR[PEG] = 0$), and the 0's that are interlocked will be merged (ORed) with 0's that are already present in the 64-bit ECC segment, unless the block is designated as an Over-Program Protected block.

Addresses in locked/disabled blocks cannot be programmed. Values may be programmed in any or all of eight words, within a page, with a single program sequence. Page-bound words have addresses which differ only in address bits [4:2]. Up to four pages can be programmed at once on a quad-page boundary, which differ only in address bits [6:5]. The program operation consists of the following sequence of events:

1. Change the value in the $MCR[PGM]$ bit from a 0 to a 1.

NOTE

Ensure the block that contains the address to be programmed is unlocked.

2. Write the first address to be programmed with the program data. The embedded flash memory latches address bits [22:7] and SoC-specific UTest NVM enable at this time. The embedded flash memory latches data written as well. This write is referred to as

a program data interlock write. An interlock write may be done as a 64-bit transaction or a 32-bit transaction. Refer to the Flash Memory Controller chapter for information on the size of writes that are allowed.

3. If more than one word or doubleword is to be programmed, write each additional address in the page with data to be programmed. This is referred to as a program data write. The embedded flash memory ignores address bits [22:7] and SoC-specific UTest NVM enable for program data writes. All unwritten data words default to 0xFFFF_FFFF.
4. Write a logic 1 to the MCR[EHV] bit to start the internal program sequence or skip to step 9 to terminate.
5. Wait until the MCR[DONE] bit goes high.

NOTE

Since MCR[DONE] clears with MCR[EHV] being set, it may not be possible for software to read MCR[DONE] as a 0 prior to this step, depending on the operation selected.

6. Confirm MCR[PEG] = 1.
7. Write a logic 0 to the MCR[EHV] bit.
8. If more addresses are to be programmed, return to step 2.
9. Write a logic 0 to the MCR[PGM] bit to terminate the program sequence.

The first write after a program is initiated determines the quad-page address to be programmed. Program may be initiated with the 0 to 1 transition of the MCR[PGM] bit or by clearing the MCR[EHV] bit at the end of a previous program. This first write is referred to as an interlock write. The interlock write determines if the UTest NVM or normal array space is to be programmed by sampling SoC-specific UTest NVM enable and causing MCR[PEAS] to be set/cleared.

In the case of an erase-suspended program, the values in MCR[PEAS] may be modified via the program interlock write, enabling erase-suspended programs to and from UTest NVM space.

An interlock write must be performed before setting MCR[EHV]. A program sequence may be terminated by clearing MCR[PGM] prior to setting MCR[EHV].

After the interlock write, additional writes affect the data to be programmed at the word location determined by address bits [4:2], as well as the page location within a 1024-bit segment (determined by address bits [6:5]). Unwritten locations default to a data value of 0xFFFF_FFFF. If multiple writes are done to the same location, the data for the last write is used in programming.

While DONE is low, EHV is high, and PSUS is low, then EHV may be cleared, resulting in a program abort. A program abort forces the embedded flash memory to step 8 of the program sequence. An aborted program results in PEG being set low, indicating a failed operation. The data space being operated on before the abort contains indeterminate data. A program sequence may not be aborted while in program suspend.

Warning

Aborting a program operation leaves the flash core addresses being programmed in an indeterminate data state. This may be recovered by executing an erase on the affected blocks.

26.6.1.2.1 Program software locking

A software mechanism is provided to independently lock/unlock blocks against program and erase.

Software locking is done through the LOCK0, LOCK1, LOCK2, and LOCK3 registers. These may be written through register writes, and may be read through register reads.

26.6.1.2.2 Program hardware locking

Hardware locking which only affects the program operation is also available for UTest, 16 KB, 32 KB, 64 KB, and 256 KB blocks.

26.6.1.2.3 Program suspend/resume

The program sequence may be suspended to allow read access to the flash core. It is not possible to erase during a program suspend, or to program during a program suspend. These operations are prevented by register locking described in the Module Configuration Register (MCR) section. Read-while-write operation may also be used to read the array during a program sequence, providing the read is to a different partition.

A program suspend can be initiated by changing the value of the MCR[PSUS] bit from a 0 to a 1. MCR[PSUS] can be set high at any time when MCR[PGM] and MCR[EHV] are high. A 0 to 1 transition of MCR[PSUS] causes the embedded flash memory to start the sequence to enter program suspend, which is a read state. MCR[DONE] must become a logic level 1 before the embedded flash memory is suspended. At this time flash core reads may be attempted. Once suspended, the flash core may only be read. During suspend, all reads to flash core locations targeted for program, and blocks targeted for erase, return indeterminate data.

The program sequence is resumed by writing a logic 0 to MCR[PSUS]. MCR[EHV] must be set to a 1 before clearing MCR[PSUS] to resume operation. When the operation resumes, the embedded flash memory continues the program sequence from one of a set of predefined points. This may extend the time required for the program operation.

NOTE

Repeated suspends at a high frequency may result in the operation not completing normally. Although suspend frequency is not limited, enabling at least 20 μ s between suspend resume and future suspend requests improve the opportunity for the flash to complete the operation successfully.

26.6.1.2.4 Over-program protection enable

One-time programming can be enabled on a block basis. In an over-program protection enabled block, any doubleword which has already been programmed cannot be programmed again. The over-program protection enable does not affect erase operation. Attempts to over-program will result in MCR[PEG] being cleared. If any doubleword within the quad-page has an over-program protection violation, MCR[PEG] will be cleared, and no doublewords will be programmed.

One-time programming can be enabled for 16 KB, 32 KB, 64 KB, or 256 KB blocks.

Over-program protection is enabled by sideband signals, and if enabled, the program operation will be modified to check for programmed bits prior to doing a high-voltage program event.

26.6.1.2.5 Successful program address

To enable the SoC to create logic to authenticate operations with a program (such as in a diary operation), the flash module provides the last successful address programmed as a sideband signal. This value will be held until the next successful program. A successful program is defined as a program operation of data that is not all 1's to an unlocked block, and the operation results in MCR[PEG] returning a 1. If any doubleword within the pages being programmed is interlocked with all 1's, the sideband signal will not be updated. An address of all 1's, and each of the block size selects selected, is the default (and invalid) address after reset.

26.6.1.3 Erase

Erase changes the value stored in all bits of the selected block(s) to logic 1. An erase sequence operates on any combination of blocks in the main address space. The erase sequence is fully automated within the flash memory. Blocks to be erased must be selected prior to initiating the erase sequence. Locked/disabled blocks cannot be erased. If multiple blocks are selected for erase during an erase sequence, the blocks are erased sequentially starting with the lowest numbered block and terminating with the highest (low first, mid second, high last, 16 KB first, 32 KB second, 64 KB third, last blocks are 256 KB). The erase sequence consists of the following events:

1. Change the value in the MCR[ERS] bit from 0 to 1.
2. Select the block or blocks to be erased by writing ones to the appropriate registers in SEL0, SEL1, SEL2, or SEL3 registers.

NOTE

Lock and Select are independent. If a block is selected and locked, no erase occurs.

3. Write to any address in flash memory. This is referred to as an erase interlock write. An erase interlock write to UTest NVM space is not allowed if sealed (since erase is not allowed in the UTest NVM space when sealed).
4. Write a logic 1 to the MCR[EHV] bit to start an internal erase sequence or skip to step 9 to terminate.
5. Wait until the MCR[DONE] bit goes high.

NOTE

Since MCR[DONE] clears with MCR[EHV] being set, it may not be possible for software to read MCR[DONE] as a 0 prior to this step, depending on the operation selected.

6. Confirm MCR[PEG] = 1.
7. Write a logic 0 to the MCR[EHV] bit.
8. If more blocks are to be erased, return to step 2.
9. Write a logic 0 to the MCR[ERS] bit to terminate the erase.

After setting ERS, one write, referred to as an interlock write, must be performed before EHV can be set to a 1. Data words written during erase sequence interlock writes are ignored. The erase sequence may be terminated by clearing ERS before setting EHV.

An erase operation may be aborted by clearing EHV, assuming DONE is low, EHV is high, and ESUS is low. An erase abort forces the embedded flash memory to step eight of the erase sequence. An aborted erase results in PEG being set low, indicating a failed operation. The block(s) being operated on before the abort contain indeterminate data. An erase sequence may not be aborted while in erase suspend.

26.6.1.3.1 Erase software locking

Software Locking affect erase operation. For details on this see [Program software locking](#).

26.6.1.3.2 Erase hardware locking

Hardware locking which only affects the erase operation is also available for UTest, 16 KB, 32 KB, 64 KB, and 256 KB blocks.

26.6.1.3.3 Erase suspend/resume

The erase sequence may be suspended to allow read access to the flash core. The erase sequence may also be suspended to program (Erase-Suspended Program) the flash core. A program started during erase suspend can in turn be suspended. Only one erase suspend and one program suspend are allowed at a time during an operation. It is not possible to erase during an erase suspend, or program during a program suspend. During suspend, all reads to flash core locations targeted for program and blocks targeted for erase return indeterminate data.

Read-while-write operation may also be used to read the array during an erase sequence, provided the read is to a partition not selected for erase.

An erase suspend can be initiated by changing the value of the MCR[ESUS] bit from a 0 to a 1. MCR[ESUS] can be set to a 1 at any time when MCR[ERS] and MCR[EHV] are high and MCR[PGM] is low. A 0 to 1 transition of MCR[ESUS] causes the embedded flash memory to start the sequence which places it in erase suspend. MCR[DONE] must become a logic level 1 before the embedded flash memory is suspended and further actions are attempted. Once suspended, the array may be read or a program sequence may be initiated (erase-suspended program). Before initiating a program sequence, MCR[EHV] must first be cleared. If a program sequence is initiated, the values of SoC-specific UTest NVM enable is recaptured. Once the erase-suspended program is completed, the value of PEAS is returned to its erase value. Flash core reads that occur while MCR[ESUS] = 1 from the block(s) being erased will return indeterminate data.

The erase sequence is resumed by writing a logic 0 to MCR[ESUS]. MCR[EHV] must be set to a 1 and MCR[PGM] must be cleared (in the event of an erase-suspended program) before MCR[ESUS] can be cleared to resume the operation. The embedded flash memory continues the erase sequence from one of a set of predefined points. This may extend the time required for the erase operation.

NOTE

1. In an erase-suspended program, programming flash core locations in blocks which were being operated on will respond by completing the operation with a fail code (MCR[PEG] = 0). Programming voltages will not be applied to the memory array in this case.
2. Repeated suspends at a high frequency may result in the operation not completing normally. Although suspend frequency is not limited, enabling at least 5ms between suspend resume and future suspend requests improve the opportunity for the flash to complete the operation successfully.

26.6.2 UTest mode

UTest mode is a mode into which customers can put the embedded flash memory so as to do specific tests that check the integrity of the embedded flash memory.

26.6.2.1 Array integrity self check

The Array Integrity (AI) operation is a flash-specific self-test feature in which a MISR signature value is generated from data read from selected flash blocks. After the operation is complete, customer code compares the resulting MISR signature to the expected value to determine if an incorrect read or ECC detection occurred.

Array integrity operations are fully contained within the flash module. Results of Array integrity operations are not reported beyond the boundary of the flash module.

During array integrity test, user mode array reads are ignored to ensure the array integrity operation is not corrupted. User mode array reads may be executed if suspended or at a breakpoint.

AI read operations are executed using a predefined address sequence selected via the UT0[AIS] register field value. The data to be read is customer-specific, e.g., user program code that has been programmed into the flash memory. Any random or nonrandom code is valid. The expected MISR signature is calculated based on the specific data to be read.

Note that array integrity MISR signature calculations are performed after ECC detection and correction so correctable ECC events do not cause the MISR signature to deviate from the expected value.

The Array integrity check consists of the following sequence:

1. Enable UTest mode by setting the UT0[UTE] bit to 1.
2. Select one or more blocks for an array integrity check by writing 1's to the appropriate bits in the SEL0 – SEL3 registers. Blocks selected for AI test do not need to be unlocked.

Flash blocks that have not been selected are still read as part of the array integrity sequence to enable full transition coverage. Reads on unselected blocks are not captured in the MISR. Selecting fewer blocks may not result in a faster array integrity execution time, depending on the combinations of blocks selected and the sequence.

Blocks protected with the Test Mode Disable Seal are still read as part of the array integrity sequence. The resulting read on sealed blocks is not captured in the MISR, but the Single Bit Correction, Double Bit Detection and breakpoints are still honored.

NOTE

It is not possible to do array integrity operations on the UTest NVM block.

3. If desired, set the UT0[AIS] bit to 1 for sequential addressing only.

For normal integrity checks of the flash memory, sequential addressing is recommended. If it is required to more fully check the read path (in a diagnostic mode), it is recommended that UT0[AIS] be left at 0, to use the address sequence that checks the read path more fully, and examine read transitions. This sequence takes more time.

4. Seed the MISR registers (UM0 — UM9) with desired values.
5. If breakpoints are desired, set UT0[AIBPE] to 1, and ensure that MCR[EER] and MCR[SBC] are cleared. If it is desired to break on single-bit correction, the UT0[SBCE] must be set.
6. Initiate AI by setting the UT0[AIE] bit to 1.

The array integrity operation may be aborted prior to UT0[AID] going high. This may be done by clearing the UT0[AIE] bit and continuing to the next step. It should be noted that in the event of an aborted array integrity check the MISR registers contain a signature for the portion of the operation that was completed prior to the abort, and are not deterministic. Prior to doing another array integrity operation, the UM0 – UM9 registers may need to be initialized to the desired seed value by doing register writes.

The array integrity operation may be suspended prior to UT0[AID] going high. UT0[AISUS] may be set to request an array integrity suspend. After the UT0[AISUS] bit is set, the user should wait for UT0[AID] bit to go high, which

indicates the flash has entered the suspend state, and normal reads to the flash may be done. After the UT0[AID] bit goes high, the UT0[AISUS] bit may be cleared to resume the array integrity sequence.

7. Wait until the UT0[AID] bit goes high, indicating that array integrity test is complete.
8. If breakpoints are enabled, check the UT0[NAIBP] bit. If set to 1, the MCR[EER], MCR[SBC], and ADDR registers may be checked to determine the cause of the break and the address of the break. Prior to resuming the operation, clear the MCR[SBC] or MCR[EER] bits, then resume the operation by clearing the UT0[NAIBP] bit. Continue to wait until the UT0[AID] bit goes high. If breakpoints are not enabled, or if the UT0[NAIBP] bit is low when the UT0[AID] bit goes high, the operation is complete. Continue to the next step.
9. Read values in the MISR registers (UM0 — UM9) to ensure correct signature.

NOTE

Array integrity reads may be done to unselected (or non-present) locations. Reads done to these locations do not update the MISR, and do not update the MCR[EER] and MCR[SBC] error bits.

10. Write a logic 0 to the UT0[AIE] bit to complete array integrity test.

26.6.2.2 User margin read

User Margin Read is a flash-specific self-test feature that enables reads to be done at a Program margin or Erase margin level.

Margin reads are performed with an internal read reference voltage that has been adjusted, i.e., "margined" toward the threshold voltage being checked. The read reference voltage is either margined toward a 0 (programmed bit) or margined towards a 1 (erased bit), depending on the margin read operation selected.

User margin read is done using the array integrity interface, and has all the associated features of the array integrity interface (MISR and Breakpoints). User margin read is a self-timed event, and is independent of system clocks or wait states selected.

Margin read operations are fully contained within the flash module. Results of margin read operations are not reported beyond the boundary of the flash module.

During margin read operations, user mode array reads are ignored to ensure the operation is not corrupted. User mode array reads may be executed if suspended or at a breakpoint.

ECC corrections and detections are noted during the user margin read test. Note that margin read MISR signature calculations are performed *after* ECC detection and correction so correctable ECC events do not cause the MISR signature to deviate from the expected value.

The data to be read is customer-specific, e.g., user program code that has been programmed into the flash memory. Any random or non-random code is valid. The expected MISR signature is calculated based on the specific data to be read. After the operation is completed the margin read results can be checked by reading the MCR[EER] and MCR[SBC] bits to determine if zero, one, or two bits are being detected by the margin read, and comparing the generated MISR signature to the expected value.

The use model for margin read is in the event of a user-detected single bit correction (through user reads). A margin read may be done to check for a possible second bit falling within the selected margin levels.

The procedure for doing margin reads is:

1. Enable UTest mode by setting the UT0[UTE] bit to 1.
2. Select one or more blocks for margin read check by writing 1's to the appropriate bits in SEL0 – SEL3 registers. Blocks selected for margin read do not need to be unlocked.

Flash blocks that have not been selected are still read as part of the margin read sequence to enable full transition coverage. Reads on unselected blocks are not captured in the MISR. Selecting fewer blocks does not result in a faster margin read execution time.

Blocks protected with the Test Mode Disable Seal are still read as part of the margin read sequence. The resulting read on sealed blocks is not captured in the MISR, but the Single Bit Correction, Double Bit Detection and breakpoints are still honored.

NOTE

It is not possible to do margin read operations on the UTest NVM block.

3. Set the UT0[AIS] bit to 1 for sequential addressing only.
4. Seed the MISR registers (UM0 — UM9) with desired values.
5. If breakpoints are desired, set UT0[AIBPE] to 1, and ensure that MCR[EER] and MCR[SBC] are cleared. If it is desired to break on single-bit correction, the UT0[SBCE] bit must be set to 1.
6. Initiate margin read by setting the UT0[MRE] bit to 1.
7. Select either one's margin or zero's margin by setting the UT0[MRV] bit to the appropriate value.
8. Set the UT0[AIE] bit.

During margin read operations, with UT0[AID] low, it is not recommended to attempt write operations to the MCR[SBC] and MCR[EER] bits. It is recommended that these bits only be written during suspend, breakpoints, or at completion of the Margin Read operation.

9. Wait until the UT0[AID] bit goes high, indicating that margin read test is complete.

If breakpoints were enabled or a suspend was requested during margin read, the operation may be at a breakpoint or a suspend state. See Step 8 in section [Array integrity self check](#) for more information.

10. Read values in the MISR registers (UM0 — UM9) to ensure correct signature.

NOTE

Margin reads may be done to unselected (or non-present) locations. Reads done to these locations do not update the MISR, and do not update the MCR[EER] and MCR[SBC] error bits.

11. Write a logic 0 to the UT0[AIE] bit.

26.7 Initialization information

A reset is the highest priority operation for the embedded flash memory and terminates all other operations.

The embedded flash memory uses reset to initialize register and status bits to their default reset values. If the embedded flash memory is executing a program or erase operation (PGM or ERS = 1) and a reset is issued, the operation is aborted and the embedded flash memory disables the high-voltage logic without damage to the high-voltage circuits. Reset aborts all operations and forces the embedded flash memory into user mode ready to receive accesses.

After reset is requested, MCR[DONE] goes low, and remains low during reset and reset recovery. At the end of reset recovery, MCR[DONE] transitions from a 0 to a 1.

After reset is completed, register reads may be done, although it should be noted that registers that require updating from UTest NVM information, or other inputs, may not read updated values until MCR[DONE] transitions high.

During reset recovery, register writes are not allowed until the MCR[DONE] bit transitions high to indicate reset recovery is completed.

Warning

Resetting during a program or erase operation leaves the flash core blocks being programmed or erased in an indeterminate data state. This may be recovered by executing an erase on the affected blocks.

Chapter 27

External Bus Interface (EBI)

27.1 Chip-specific EBI information

The EBI is not available on the BGA416 package.

27.1.1 EBI use cases

The following table summarizes the primary use cases of the EBI on this chip.

Table 27-1. EBI use cases

Use case	EBI configuration
EBI used for external boot	Selected when the SIU_RSR[BOOTCFG] value of 10b or 11b determines the boot configuration. In this case, the Boot Assist Module (BAM) configures the SIU's applicable PCRs and the EBI registers according to the 10b or 11b value of SIU_RSR[BOOTCFG] and the value of RCHW[PS0]. For details, see the Functional Description of the BAM. User code can modify the SIU's applicable PCRs and EBI registers at any time.
EBI not used for external boot	User code must configure SIU's applicable PCRs and the EBI registers to select the appropriate mode.

27.1.2 EBI signal names

The following table aligns the chip-level pin/ball names for EBI signals with the module-level signal names.

For detailed EBI signal descriptions that use the module-level signal names, see [External Signal Description](#).

Table 27-2. EBI signals: chip-to-module alignment

Chip-level pin/ball name	Chip-level I/O	Module-level signal name	Note
D_ADD[9:30]	Output	ADDR[3:31]	ADDR[31] maps to $\overline{D\_CS}[2]$. $\overline{D\_CS}[2]$ is the primary function. The secondary function is EBI data only in non-muxed mode and address/data in muxed mode.
$\overline{D\_BDIP}$	Output	$\overline{BDIP}$	—
D_CLKOUT	Output	CLKOUT	—
$\overline{D\_CS}[0:3]$	Output	$\overline{CAL\_CS}[0:3]$	$\overline{D\_CS}[2]$ maps to ADDR[31]. $\overline{D\_CS}[2]$ is the primary function. The secondary function is EBI data only in non-muxed mode and address/data in muxed mode.
D_ADD_DAT[0:31]	Input/output	DATA[0:31]	In non-multiplexed mode, D_ADD_DAT[0:15] provide data only.
$\overline{D\_OE}$	Output	$\overline{OE}$	—
D_RD_WR	Output	RD_WR	—
$\overline{D\_TA}$	Input/output	$\overline{TA}$	—
$\overline{D\_TEA}$	Input/output	$\overline{TEA}$	—
$\overline{D\_TS}$	Output	$\overline{TS}$	—
$\overline{D\_WE}[0:3]$	Output	$\overline{WE}[0:3]/\overline{BE}[0:3]$	—
D_ALE	Output	—	When asserted, the Address Latch Enable (D_ALE) signal indicates that a valid address is present during a multiplexed bus access using a D_ADD_DAT signal. D_ALE can be used in conjunction with an external latch to hold the state of the address access when connecting the MCU to a non-multiplexed bus compatible memory.

27.1.3 EBI supported configurations

The following table shows the function of the external pins in each of the possible muxed/non-muxed usage configurations allowed for this device.

In any muxed mode, an external latch is needed to latch address [16:30] in 16-bit mode or address [16:29] in 32-bit mode. The latch must be connected to D_ADD_DAT pins.

Table 27-3. Function of EBI pins for supported configurations

Mode	$\overline{D\_WE}[0:3]$	D_ADD[9:15]	D_ADD[16:30] ¹ D_ADD_DAT[16:30] ²	D_ADD_DAT [0:15]	D_CS2 ³
Non-muxed 16-bit mode	write/byte enable [0:1]	Address [9:15]	Address [16:30]	Data [0:15]	Address [31]
Muxed 32-bit mode	write/byte enable [0:3]	Address [9:15]	Address [16:30] / Data [16:30]	Address [0:15] / Data [0:15]	Address [31] / Data [31]
Muxed 16-bit mode: EBI_MCR[D16_31] is 1	write/byte enable [0:1]	Address [9:15]	Address [16:30] / Data [0:14]	Not used	Address [31] / Data [15]
Muxed 16-bit mode: EBI_MCR[D16_31] is 0	write/byte enable [0:1]	Address [9:15]	Not used	Address [16:31] / Data [0:15]	Not used

1. D_ADD[16:30] SIU PCR functionality must be selected in non-multiplexed mode: AD_MUX is 0.

2. D_ADD_DAT[16:30] SIU PCR functionality must be selected in multiplexed mode: AD_MUX is 1.
3. D_CS2 is the primary function. The secondary function is EBI data only in non-muxed mode and address/data in muxed mode.

27.1.4 EBI CLKOUT division for slower-speed modes

[Slower-Speed Modes](#) explains that the external CLKOUT frequency can be divided by writing a clock control register in a block outside the EBI. On this chip, the register field SIU_ECCR[EBDF] enables CLKOUT division by a programmable value from 1 to 4.

27.1.5 EBI register reset values

The following table identifies chip-specific reset values of EBI registers.

Table 27-4. EBI register chip-specific reset values

Register	Reset value	Notes
MCR	0000_0800h	The DBM and D16_31 fields both reset to 0.
CAL_BR n	2000_0002h	The bits BA[0:2] have the fixed value 001b.
CAL_OR n	E000_0000h	The bits AM[0:2] have the fixed value 111b.

27.1.6 EBI_TESR in Module Disable mode

When EBI_MCR[MDIS] is 1 so that the EBI is in Module Disable mode, EBI_TESR cannot be written.

27.2 Introduction

27.2.1 Overview

The External Bus Interface (EBI) handles the transfer of information between the internal buses and the memories or peripherals in the external address space. The EBI includes a memory controller that generates interface signals to support a variety of external memories. This includes Single Data Rate (SDR) burst mode flash, SRAM, and asynchronous memories. It supports up to four regions (via chip selects), each with its own programmed attributes (plus up to four calibration chip-select regions).

27.2.2 Features

Note

This list is a superset list of all possible features the EBI supports. See the chip-specific EBI information for details on specifics for a particular device due to package limitations.

- 32-Bit Address bus with transfer size indication
- 32-Bit Data bus (16-bit Data Bus Mode also supported)
- Multiplexed Address on Data pins (single master)
- Memory controller with support for various memory types:
 - synchronous burst SDR flash and SRAM
 - asynchronous/legacy flash and SRAM
- Burst support (wrapped only)
- Bus monitor
- Port size configuration per chip select (16 or 32 bits)
- Configurable wait states
- Configurable internal or external transfer acknowledge ($\overline{TA}$) per chip select
- Support for Dynamic Calibration with up to four chip selects
- Four Write/Byte Enable ($\overline{WE}[0:3]/\overline{BE}[0:3]$) signals
- Slower-speed clock modes
- Stop and Module Disable Modes for power savings
- Optional automatic CLKOUT gating to save power and reduce EMI
- Misaligned access support (for chip-select accesses only)
- Compatible with MPC5xx external bus (with some limitations)

27.2.3 Modes of Operation

The mode of the EBI is determined by the MDIS, DBM, and AD_MUX bits in the EBI_MCR. See [EBI Module Configuration Register \(EBI_MCR\)](#) for details. Slower-speed modes, Debug Mode, Stop Mode, and Factory Test Mode are modes that the MCU may enter, in parallel to the EBI being configured in one of its block-specific modes.

27.2.3.1 Single Master Mode

In Single Master Mode, the EBI responds to internal requests matching one of its regions, but ignores all externally-initiated bus requests. The MCU is the only master allowed to initiate transactions on the external bus in this mode; therefore, it acts as a parked master and does not have to arbitrate for the bus before starting each cycle. Single Master Mode is entered when MDIS=0 in the EBI_MCR.

27.2.3.2 Module Disable Mode

The Module Disable Mode is used for MCU power management. The clock to the non-memory mapped logic in the EBI can be stopped while in Module Disable Mode. Logic on the MCU external to the EBI is needed to fully implement the Module Disable Mode (to shut off the clock). Internal master requests made to the external bus in Module Disable Mode are terminated with transfer error (internally, no external $\overline{\text{TEA}}$ assertion). Module Disable Mode is entered when MDIS=1 in the EBI_MCR.

27.2.3.3 Stop Mode

The EBI supports the IPI Green-Line Interface Stop Mode mechanism used for MCU power management. When a request is made to enter Stop Mode (controlled in device logic outside EBI), the EBI block completes any pending bus transactions and acknowledges the stop request. After the acknowledgement, the system clock input may be shut off by the clock driver on the MCU. While the clocks are shut off, the EBI is not accessible. While in stop mode, accesses to the EBI from the internal master will terminate with transfer error (internally, no external $\overline{\text{TEA}}$ assertion).

27.2.3.4 Slower-Speed Modes

In slower-speed modes, the external CLKOUT frequency is divided (by 2, 3, etc.) compared with that of the internal system bus. The EBI behavior remains dictated by the mode of the EBI, except that it drives and samples signals at the CLKOUT frequency

rather than the internal system frequency. This mode is selected by writing a clock control register in a block outside of the EBI. Refer to the device-specific documentation to see which slower-speed modes are available for a particular MCU (1/2, 1/3, etc.).

27.2.3.5 16-Bit Data Bus Mode

For MCUs that have only 16 data bus signals pinned out, or for systems where the use of a different multiplexed function (e.g. GPIO) is desired on 16 of the 32 data pins, the EBI supports a 16-bit Data Bus Mode. In this mode, only 16 data signals are used by the EBI. The user can select which 16 data signals are used (DATA[0:15] or DATA[16:31]) by writing the D16_31 bit in the EBI_MCR.

For EBI-mastered accesses, the operation in 16-bit Data Bus Mode (DBM=1, PS=x) is similar to a chip-select access to a 16-bit port in 32-bit Data Bus Mode (DBM=0, PS=1), except for the case of a non-chip-select access of exactly 32-bit size.

EBI-mastered non-chip-select accesses of exactly 32-bit size are supported via a two (16-bit) beat burst for both reads and writes. See [Non-Chip-Select Burst in 16-bit Data Bus Mode](#). Non-chip-select transfers of non-32-bit size are supported in standard non-burst fashion.

16-bit Data Bus Mode is entered when DBM=1 in the EBI_MCR. Some MCUs may have DBM=1 by default out of reset. See the device-specific documentation for the DBM and D16_31 reset values.

27.2.3.6 Multiplexed Address on Data Bus Mode

This mode covers several cases aimed at reducing pin count on MCU and external components. In this mode, the DATA pins will drive (for internal master cycles) the address value on the first clock of the cycle (while $\overline{TS}$ is asserted).

The memory controller supports per-chip-select selection of multiplexing address/data through the BRx[AD_MUX] bit.

Address on Data bus multiplexing also supports the 16-bit data bus mode (MCR[DBM]=1) and 16-bit memories (ORx[PS]=1). The user can select which 16 data signals are used (DATA[0:15] or DATA[16:31]) by writing the D16_31 bit in the EBI_MCR. For either setting of D16_31, the 16 LSBs of external address (ADDR[16:31]) are driven onto the selected 16 DATA pins. If additional address lines are required to interface to the memory, then non-muxed address pins are sometimes (see note below) required to complete the address space (e.g. ADDR[8:15] are commonly present as non-muxed address pins).

Note

The EBI also drives the unused 16 DATA signals with the MSBs of the external address, zero-padded in front (e.g. when D16_31 bit is set for a device with 24 ADDR pins, the EBI drives (0b00000000, ADDR[8:15]) on DATA[0:15]. This allows the device to optionally use DATA[8:15] for the upper 8 external address lines instead of requiring separate non-muxed ADDR[8:15] pins. This is relevant primarily for devices that support both 32-bit and 16-bit A/D muxed operation, so therefore have DATA[0:31] pins present on the device, and in that case are not required to have separate ADDR pins.

For more details (e.g. timing diagrams), see [Address Data Multiplexing](#).

27.2.3.7 Debug Mode

When the MCU is in Debug Mode, the EBI behavior is unaffected and remains dictated by the mode of the EBI.

27.3 External Signal Description

27.3.1 Overview

The following table lists the external pins used by the EBI. Not all signals listed here are available external to the chip.

Table 27-5. Signal Properties

Name	I/O Type	Function	Pull ¹
ADDR[3:31]	I/O	Address bus	-
BDIP	Output	Burst Data in Progress	Up
CLKOUT ²	Output	Clockout	-
CAL_CS[0:3]	Output	Calibration Chip Selects	Up
DATA[0:31]	I/O	Data bus ³	-
OE	Output	Output Enable	Up
RD_WR	I/O	Read_Write	Up
TA	I/O	Transfer Acknowledge	Up
TEA	I/O	Transfer Error Acknowledge	Up

Table continues on the next page...

Table 27-5. Signal Properties (continued)

Name	I/O Type	Function	Pull ¹
TS	I/O	Transfer Start	Up
WE[0:3]/BE[0:3]	Output	Write/Byte Enables	Up

1. This column shows which signals require a weak pullup or pulldown. The EBI block does not contain these pullup/pulldown devices within the block. They are assumed to be in another module of the MCU (e.g. pads module).
2. The CLKOUT signal is driven by the System Clock Block outside the EBI.
3. In Address/Data multiplexing modes, Data will also show the address during the address phase.

27.3.2 Detailed Signal Descriptions

Note

This section lists the superset of signals for the EBI. Refer to the chip-specific EBI information for package limitations and possible signal renaming.

27.3.2.1 ADDR [3:31] — Address Lines 3-31

The ADDR[3:31] signals specify the physical address of the bus transaction.

The ADDR[3:31] address lines correspond to bits 3-31 of the EBI's 32-bit internal address bus.

27.3.2.2 $\overline{\text{BDIP}}$ — Burst Data in Progress

$\overline{\text{BDIP}}$ is asserted to indicate that the master is requesting another data beat following the current one.

This signal is driven by the EBI on all EBI-mastered external burst cycles, but is only sampled by burst mode memories that have a corresponding pin. See [Burst Transfer](#).

27.3.2.3 CLKOUT — Clockout

CLKOUT is a general-purpose clock output signal to connect to the clock input of SDR external memories and in some cases to the input clock of another MCU in multi-master configurations.

27.3.2.4 $\overline{\text{CAL\_CS}}$ [0:3] — Calibration Chip Selects 0-3

$\overline{\text{CAL\_CS}}_x$ is asserted by the master to indicate that this transaction is targeted for a particular memory bank on the Calibration external bus.

The calibration chip selects are driven only by the EBI. External master accesses on the Calibration bus are not supported. In all other aspects, the calibration chip-selects behave exactly as the primary chip-selects. See [Memory Controller with Support for Various Memory Types](#) for details on chip-select operation.

27.3.2.5 DATA [0:31] — Data Lines 0-31

The DATA[0:31] signals contain the data to be transferred for the current transaction.

DATA[0:31] is driven by the EBI when it owns the external bus and it initiates a write transaction to an external device.

DATA[0:31] is driven by an external device during a read transaction from the EBI.

For 8-bit and 16-bit transactions, the byte lanes not selected for the transfer do not supply valid data.

DATA[0:31] is driven by the EBI in the address phase with the ADDR value if the Address on Data multiplexing mode is enabled. See [Multiplexed Address on Data Bus Mode](#) for details.

In 16-bit Data Bus Mode, (or for chip-select accesses to a 16-bit port), only DATA[0:15] or DATA[16:31] are used by the EBI, depending on the setting of the D16_31 bit in the EBI_MCR. See [16-Bit Data Bus Mode](#).

27.3.2.6 $\overline{\text{OE}}$ — Output Enable

$\overline{\text{OE}}$ is used to indicate when an external memory is permitted to drive back read data. External memories must have their data output buffers off when $\overline{\text{OE}}$ is negated. $\overline{\text{OE}}$ is only asserted for chip-select accesses.

For read cycles, $\overline{\text{OE}}$ is asserted one clock after $\overline{\text{TS}}$ assertion (by default with EBI_BR[EOE]=0b00) and held until the termination of the transfer. For write cycles, $\overline{\text{OE}}$ is negated throughout the cycle.

27.3.2.7 RD_ $\overline{\text{WR}}$ — Read / Write

RD_ $\overline{\text{WR}}$ indicates whether the current transaction is a read access or a write access.

$\overline{RD\_WR}$ is driven in the same clock as the assertion of $\overline{TS}$ and valid address, and is kept valid until the cycle is terminated.

27.3.2.8 $\overline{TA}$ — Transfer Acknowledge

$\overline{TA}$ is asserted to indicate that the slave has received the data (and completed the access) for a write cycle, or returned data for a read cycle. If the transaction is a burst read, $\overline{TA}$ is asserted for each one of the transaction beats. For write transactions, $\overline{TA}$ is only asserted once at access completion, even if more than one write data beat is transferred.

$\overline{TA}$ is driven by the EBI when the access is controlled by the chip selects (and $SETA=0$). Otherwise, $\overline{TA}$ is driven by the slave device to which the current transaction was addressed.

See [Termination Signals Protocol](#) for more details.

27.3.2.9 $\overline{TEA}$ — Transfer Error Acknowledge

$\overline{TEA}$ is asserted by either the EBI or an external device to indicate that an error condition has occurred during the bus cycle.

$\overline{TEA}$ is asserted by the EBI when the internal bus monitor detected a timeout error.

See [Termination Signals Protocol](#) for more details.

27.3.2.10 $\overline{TS}$ — Transfer Start

$\overline{TS}$ is asserted by the current bus owner to indicate the start of a transaction on the external bus.

$\overline{TS}$ is only asserted for the first clock cycle of the transaction, and is negated in the successive clock cycles until the end of the transaction.

27.3.2.11 $\overline{WE}[0:3] / \overline{BE}[0:3]$ — Write/Byte Enables 0-3

Write enables are used to enable program operations to a particular memory. These signals can also be used as byte enables for read and write operation by setting the WEBS bit in the appropriate Base Register. $\overline{WE}[0:3]/\overline{BE}[0:3]$ are only asserted for chip-select accesses.

For chip-select accesses to a 16-bit port, only $\overline{WE}[0:1]/\overline{BE}[0:1]$ are used by the EBI, regardless of which half of the DATA bus is selected via the D16_31 bit in the EBI_MCR.

See [Four Write/Byte Enable \(\$\overline{WE}/\overline{BE}\$ \) Signals](#) for more details on $\overline{WE}[0:3]/\overline{BE}[0:3]$ functionality.

27.3.3 Signal Output Buffer Enable Logic by Mode

The following table describes how the EBI drives its output buffer enable (OBE) signals. These are internal signals from the EBI to device logic outside the EBI, that determine when the EBI strongly drives values on pins. When the OBE for an EBI signal is asserted (1), the EBI strongly drives the value on that pin. When the OBE is negated (0), the EBI does not drive the signal, and the value is determined by internal or external pullups/pulldowns, and/or device logic outside EBI block. The logic in the following table can be overwritten by device logic, so see the device-specific documentation for any exceptions to the logic below.

NOTE

In the table below, Module Disable Mode assumes that the clock to the EBI is shut off when MDIS=1. This is an optional device feature. If the clocks are left running to EBI even when MDIS=1, then the EBI OBE behavior is as if in Single Master Mode (though EBI accesses are not supported in this scenario).

Signal	Module Disable Mode (MDIS=1)	Single Master Mode (MDIS=0)
ADDR[3:31]	0	1
BDIP	0	1
CAL_CS[0:3]	0	1
DATA[0:31]	0	Only 1 during write access or on Address phase when Addr/Data muxing is enabled.
$\overline{OE}$	0	1
RD_W $\overline{R}$	0	1
TA	0	Only 1 during chip-select (or cal-chip-select) SETA=0 access
TEA	0	Only 1 for 2 cycles when timeout occurs
TS	0	1
WE[0:3]/BE[0:3]	0	1

27.4 Memory map and register definition

This section provides a detailed description of all registers accessible in this module.

The following table gives an overview of the module registers implemented.

NOTE

Other than the exceptions noted below, EBI registers must not be written while a transaction to the EBI is in progress (or within 2 CLKOUT cycles after a transaction has just completed, to allow internal state machines to go IDLE). In those cases, the behavior is undefined.

Exceptions that can be written while an EBI transaction is in progress:

- All bits in EBI_TESR

See [Bootling from External Memory](#) for related application information.

EBI memory map

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
0	EBI Module Configuration Register (EBI_MCR)	32	R/W	See section	27.4.1/973
8	EBI Transfer Error Status Register (EBI_TESR)	32	R/W	0000_0000h	27.4.2/975
C	EBI Bus Monitor Control Register (EBI_BMCR)	32	R/W	See section	27.4.3/976
40	EBI Calibration Base Register (EBI_CAL_BR0)	32	R/W	See section	27.4.4/977
44	EBI Calibration Option Register (EBI_CAL_OR0)	32	R/W	See section	27.4.5/980
48	EBI Calibration Base Register (EBI_CAL_BR1)	32	R/W	See section	27.4.4/977
4C	EBI Calibration Option Register (EBI_CAL_OR1)	32	R/W	See section	27.4.5/980
50	EBI Calibration Base Register (EBI_CAL_BR2)	32	R/W	See section	27.4.4/977
54	EBI Calibration Option Register (EBI_CAL_OR2)	32	R/W	See section	27.4.5/980
58	EBI Calibration Base Register (EBI_CAL_BR3)	32	R/W	See section	27.4.4/977
5C	EBI Calibration Option Register (EBI_CAL_OR3)	32	R/W	See section	27.4.5/980

27.4.1 EBI Module Configuration Register (EBI_MCR)

The EBI Module Configuration Register contains bits that configure various attributes associated with EBI operation.

Address: 0h base + 0h offset = 0h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15		
R	0					0			0									
W																		
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0		
Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31		
R	ACGE	0					0				MDIS	0				D16_31	AD_MUX	DBM
W																		
Reset	0	0	0	0	1	0	0	0	0	0	0	0	0	*	0	*		

* Notes:

- DBM field: Reset value is device-specific. Refer to the device-specific documentation for the reset value for a particular MCU.
- D16_31 field: Reset value is device-specific. Refer to the device-specific documentation for the reset value for a particular MCU.

EBI_MCR field descriptions

Field	Description
0–4 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
5–7 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
8–15 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
16 ACGE	Automatic CLKOUT Gating Enable The ACGE bit enables the EBI feature of turning off CLKOUT (holding it high) during idle periods in-between external bus accesses. 0 Automatic CLKOUT Gating is disabled 1 Automatic CLKOUT Gating is enabled
17–20 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
21–24 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

Table continues on the next page...

EBI_MCR field descriptions (continued)

Field	Description
25 MDIS	Module Disable Mode The MDIS bit controls an internal EBI "enable clk" signal which can be used (if MCU logic supports) to control the clocks to the EBI. The MDIS bit allows the clock to be stopped to the non-memory mapped logic in the EBI, effectively putting the EBI in a software controlled power-saving state. See Module Disable Mode for more information. No external bus accesses can be performed when the EBI is in Module Disable Mode (MDIS=1). 0 Module Disable Mode is inactive (assert "enable clk" signal) 1 Module Disable Mode is active (negate "enable clk" signal)
26–28 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
29 D16_31	Data Bus 16_31 Select The D16_31 bit controls whether the EBI uses the DATA[0:15] or DATA[16:31] signals, when in 16-bit Data Bus Mode (DBM=1) or for chip-select accesses to a 16-bit port (PS=1). NOTE: There is one usage case where D16_31=0 cannot be used. For systems that are using both a 32-bit non-chip-select device AND a 16-bit device (chip-select or non-chip-select), the user must set D16_31=1. Otherwise (if D16_31=0 is used), when an access to the 16-bit device is performed, the 32-bit non-chip-select device may decode the wrong upper address bits and may respond to that access, causing contention on the shared address/data lines going to both devices. This is due to the 32-bit device expecting address 8:15 on DATA[8:15] pins, whereas D16_31=0 has address 24:31 on those pins. This contention case is avoided when D16_31=1, because the shared address/data pins have the same functions for both the 16-bit and 32-bit memory accesses (address 16:31 on DATA[16:31] pins). This contention case is also avoided when only chip-select devices are used, since the devices ignore accesses when their corresponding chip-select is not asserted. 0 DATA[0:15] signals are used for 16-bit port accesses 1 DATA[16:31] signals are used for 16-bit port accesses
30 AD_MUX	Address on Data Bus Multiplexing Mode The AD_MUX bit controls whether non-chip-select accesses have the address driven on the data bus in the address phase of a cycle. 0 Only Data on data pins for non-CS accesses. 1 Address on Data Multiplexing Mode is used for non-CS accesses.
31 DBM	Data Bus Mode The DBM bit controls whether the EBI is in 32-bit or 16-bit Data Bus Mode. 0 32-bit Data Bus Mode is used 1 16-bit Data Bus Mode is used

27.4.2 EBI Transfer Error Status Register (EBI_TESR)

The EBI Transfer Error Status Register contains a bit for each type of transfer error on the external bus. A bit set to logic 1 indicates what type of transfer error occurred since the last time the bits were cleared. Each bit can be cleared by reset or by writing a 1 to it. Writing a 0 has no effect. For some MCUs with this EBI, this register may not be writable in Module Disable Mode due to the use of power saving clock modes. Refer to the device-specific documentation for details.

Address: 0h base + 8h offset = 8h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15															
R	0																														
W																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0															
Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31															
R	0																													TEAF	BMTF
W																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0															

EBI_TESR field descriptions

Field	Description
0–29 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
30 TEAF	Transfer Error Acknowledge Flag This bit is set if the cycle was terminated by an externally generated TEA signal. 0 No error 1 External TEA occurred

Table continues on the next page...

EBI_TESR field descriptions (continued)

Field	Description
31 BMTF	Bus Monitor Timeout Flag This bit is set if the cycle was terminated by a bus monitor timeout. 0 No error 1 Bus monitor timeout occurred

27.4.3 EBI Bus Monitor Control Register (EBI_BMCR)

The EBI Bus Monitor Control Register controls the timeout period of the bus monitor and whether it is enabled or disabled.

Address: 0h base + Ch offset = Ch

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	
R	BMT								BME	0							
W																	
Reset	1	1	1	1	1	1	1	1	1	0	0	0	0	0	0	0	

EBI_BMCR field descriptions

Field	Description
0–15 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
16–23 BMT	Bus Monitor Timing This field defines the timeout period, in 8 external bus clock resolution, for the Bus Monitor. See Bus Monitor for more details on bus monitor operation. Timeout Period = (2 + (8 * BMT)) / external bus clock frequency.
24 BME	Bus Monitor Enable This bit controls whether the bus monitor is enabled for internal to external bus cycles. The BME bit is ignored (treated as 0) for chip-select accesses with internal TA (SETA=0). 0 Disable bus monitor 1 Enable bus monitor (for external TA accesses only)
25–31 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

27.4.4 EBI Calibration Base Register (EBI_CAL_BRn)

The EBI Base Registers are used to define the base address and other attributes for the corresponding chip select.

NOTE

An MCU may have some of the upper bits of the BA field tied to a fixed value internally in order to restrict the address range of the EBI for that MCU. Refer to the device-specific documentation to see which bits are tied off, if any, for a particular MCU. Tied-off bits can be read but not written. These bits are ignored by the EBI during the chip-select address comparison. However, the internal bridge of the MCU most likely requires that the chip-select banks be located in memory regions corresponding to the fixed values chosen.

Table 27-11. EOE Values

Value	Timing of OE Assertion
0b00 ¹	Assert OE 1 CLKOUT cycle after TS assertion (see Figure 27-17)
0b01	Assert OE the same CLKOUT cycle as TS assertion (see Figure 27-20)
0b10	Assert OE one internal clock cycle after TS assertion (see Figure 27-20) ²
0b11	Reserved

1. The 0b00 timing case also applies when EBI_BR[ADMUX]=1, regardless of EOE value in Base Register.
2. In divided bus speed modes (where the EBI runs at a slower frequency than the internal system bus), the EBI uses the internal higher-frequency clock (ipg_clk) to assert OE partway through the CLKOUT cycle where TS is asserted. In 1:1 bus speed mode (where CLKOUT and internal system bus run at same frequency), the behavior for EOE Value 0b10 is identical to that of Value 0b00.

Table 27-12. SBL, BL Values

Value		Burst Length ¹	PS	# Beats in Burst ²
SBL	BL			
0	0 ³	8-word ⁴	0 (32-bit)	8
			1 (16-bit)	16
0	1	4-word	0 (32-bit)	4
			1 (16-bit)	8
1	X	2 word ⁴	0 (32-bit)	2
			1 (16-bit)	4

1. Total amount of data fetched in a burst transfer, measured in 32-bit words.
2. Number of external data beats used in external burst transfer. The size of each beat is determined by PS value.
3. An 8-word burst length is only supported for devices using 64-bit AMBA data bus width to EBI.

Memory map and register definition

4. A word always refers to 32-bits of data, regardless of PS.

Address: $0h \text{ base} + 40h \text{ offset} + (8d \times i)$, where $i=0d$ to $3d$

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	BA															
W																
Reset	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*
Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	BA	0		LWRN	PS	EOE		SBL	AD_MUX	BL	WEBS	TBDIP	GCSN	SETA	BI	V
W																
Reset	0*	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0

* Notes:

- BA field: Some upper bits of the BA field may be tied to a fixed value, in which case the reset value is this fixed value and not zero. See the chip-specific EBI information to see which bits this applies to, if any.

EBI_CAL_BRn field descriptions

Field	Description
0–16 BA	Base Address These bits are compared to the corresponding unmasked address signals among ADDR[0:16] of the internal address bus to determine if a memory bank controlled by the memory controller is being accessed by an internal bus master. Some upper bits of the BA field may be tied to a fixed value, in which case the reset value is this fixed value and not zero. See the chip-specific EBI information to see which bits this applies to, if any.
17–18 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
19 LWRN	Late RD_W$\overline{R}$ Negation The LWRN bit determines the timing of RD_W$\overline{R}$ signal negation for a write transfer (except for SETA=1 transfers, where RD_W$\overline{R}$ negates with $\overline{CS}$ by default, so LWRN has no effect). 0 Negate (set up to high level) RD_W$\overline{R}$ one CLKOUT cycle earlier than $\overline{CS}$ negation. See Figure 27-22. 1 Negate (set up to high level) RD_W$\overline{R}$ the same CLKOUT cycle as $\overline{CS}$ negation. See Figure 27-26.
20 PS	Port Size The PS bit determines the data bus width of transactions to this chip-select bank. NOTE: In the case where the DBM bit in EBI_MCR is set for 16-bit Data Bus Mode, the PS bit value is ignored and is always treated as a '1' (16-bit port). 0 32-bit port 1 16-bit port
21–22 EOE	Early OE The EOE field determines the timing of OE signal assertion for a read transfer. When EBI_BR[ADMUX]=1, the EOE value is ignored and treated as 0b00 (in order to avoid contention on shared address/data bus for muxed transfers).

Table continues on the next page...

EBI_CAL_BRn field descriptions (continued)

Field	Description
23 SBL	Short Burst Length The SBL bit provides support for a 2-word external burst (see Figure 27-46 in Small Access Example #5: 32-byte Read to 32-bit Port with SBL=1). When SBL=1, the number of beats in a burst is automatically determined by the EBI to be 2 or 4 according to the Port Size (PS bit), regardless of BL bit value. When SBL=0, the number of beats in a burst is determined by the BL bit. See Table 27-12 under BL bit description for SBL and BL encodings.
24 AD_MUX	Address on Data Bus Multiplexing The AD_MUX bit controls whether accesses for this chip select have the address driven on the data bus in the address phase of a cycle 0 Address on Data Multiplexing Mode is disabled for this chip select. 1 Address on Data Multiplexing Mode is enabled for this chip select.
25 BL	Burst Length The BL bit (along with SBL bit) determines the amount of data transferred in a burst for this chip select, measured in 32-bit words. When SBL=0, the number of beats in a burst is automatically determined by the EBI to be 4, 8, or 16 according to the Port Size (PS bit) so that the burst fetches the number of words chosen by BL. For internal AMBA data bus width of 32-bits, the BL bit is ignored (treated as 1). When SBL=1, the BL bit value is a don't care. See Table 27-12 for SBL and BL encodings.
26 WEBS	Write Enable / Byte Select This bit controls the functionality of the WE[0:3]/BE[0:3] signals. 0 The WE[0:3]/BE[0:3] signals function as WE[0:3] 1 The WE[0:3]/BE[0:3] signals function as BE[0:3]
27 TBDIP	Toggle Burst Data in Progress This bit determines how long the $\overline{\text{BDIP}}$ signal is asserted for each data beat in a burst cycle. See TBDIP Effect on Burst Transfer for details. 0 Assert $\overline{\text{BDIP}}$ throughout the burst cycle, regardless of wait state configuration 1 Only assert $\overline{\text{BDIP}}$ (BSCY+1) external cycles before expecting subsequent burst data beats
28 GCSN	Gaurantee CS Negation The GCSN bit allows support for gauranteeing that the EBI will negate CS between all back-to-back transfer cases (even those that are part of a set of Small Accesses). See Figure 27-31. 0 Default operation (CS may or may not negate between transfers, depending on the particular back-to-back case). 1 Negate CS between all external transfers. This adds an extra dead cycle for some back-to-back cases.
29 SETA	Select External Transfer Acknowledge The SETA bit controls whether accesses for this chip select will terminate (end transfer without error) based on externally asserted TA or internally asserted TA. SETA should only be set when the BI bit is 1 as well, since burst accesses with SETA=1 are not supported. Setting SETA=1 causes the BI bit to be ignored (treated as 1, burst inhibited). 0 Transfer Acknowledge (TA) is an output from the EBI, data phase will be terminated by the EBI 1 Transfer Acknowledge (TA) is an input to the EBI, data phase will be terminated by an external device
30 BI	Burst Inhibit This bit determines whether or not burst read accesses are allowed for this chip-select bank. The BI bit is ignored (treated as 1) for chip-select accesses with external TA (SETA=1).

Table continues on the next page...

EBI_CAL_BRn field descriptions (continued)

Field	Description
	0 Enable burst accesses for this bank 1 Disable burst accesses for this bank . This is the default value out of reset (or when SETA=1).
31 V	Valid bit The user writes this bit to indicate that the contents of this Base Register and Option Register pair are valid. The appropriate CS signal does not assert unless the corresponding V-bit is set. 0 This bank is not valid 1 This bank is valid

2. The 0b00 timing case also applies when EBI_BR[ADMUX]=1, regardless of EOE value in Base Register.
3. In divided bus speed modes (where the EBI runs at a slower frequency than the internal system bus), the EBI uses the internal higher-frequency clock (ipg_clk) to assert OE partway through the CLKOUT cycle where TS is asserted. In 1:1 bus speed mode (where CLKOUT and internal system bus run at same frequency), the behavior for EOE Value 0b10 is identical to that of Value 0b00.
4. Total amount of data fetched in a burst transfer, measured in 32-bit words.
5. Number of external data beats used in external burst transfer. The size of each beat is determined by PS value.
6. An 8-word burst length is only supported for devices using 64-bit AMBA data bus width to EBI.
7. A word always refers to 32-bits of data, regardless of PS.

27.4.5 EBI Calibration Option Register (EBI_CAL_ORn)

The EBI Option Registers are used to define the address mask and other attributes for the corresponding chip select.

NOTE

An MCU may have some of the upper bits of the AM field tied to a fixed value internally in order to restrict the address range of the EBI for that MCU. See the corresponding Note for the Base Register BA field for more details. Refer to the device-specific documentation to see which bits are tied off, if any, for a particular MCU. Tied-off bits can be read but not written.

Table 27-17. BSCY Values

Value	Meaning
00	0-clock cycle wait states (1 clock per data beat)
01	1-clock cycle wait states (2 clocks per data beat)
10	2-clock cycle wait states (3 clocks per data beat)
11	3-clock cycle wait states (4 clocks per data beat)

Address: $0h \text{ base} + 44h \text{ offset} + (8d \times i)$, where $i=0d$ to $3d$

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	AM															
W																
Reset	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*
Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	AM	0	APS	0	AWE	0	SCY						0	BSCY		0
W																
Reset	0*	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

* Notes:

- AM field: Some upper bits of the AM field may be tied to a fixed value, in which case the reset value is this fixed value and not zero. See the chip-specific EBI information to see which bits this applies to, if any.

EBI_CAL_ORn field descriptions

Field	Description
0–16 AM	Address Mask This field allows masking of any corresponding bits in the associated Base Register . Masking the address independently allows external devices of different size address ranges to be used . Any clear bit masks the corresponding address bit . Any set bit causes the corresponding address bit to be used in comparison with the address pins. Address mask bits can be set or cleared in any order in the field, allowing a resource to reside in more than one area of the address map. This field can be read or written at any time. Some upper bits of the AM field may be tied to a fixed value, in which case the reset value is this fixed value and not zero. See the chip-specific EBI information to see which bits this applies to, if any.
17–18 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
19 APS	Address by Port Size The user writes this bit to cause the EBI to shift the address it drives out for accesses to this chip-select, such that the lowest address bit addresses an element of data that matches the port size specified in the corresponding EBI_BR[PS] field. APS=0, PS=X: Default Byte Addressing APS=1, PS=0: Word Addressing: EBI drives internal address 6:29 to ADDR[8:31] APS=1, PS=1: Half-Word Addressing: EBI drives internal address 7:30 to ADDR[8:31] See Address by Port Size for additional details.
20 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
21 AWE	Address / Write Enable Select The user writes this bit to cause the EBI to drive its internal WE[0:3] onto the upper 4 address bus signals (such as in systems where upper 4 address bus signals are not needed and separate WE[0:3] pins are not available). AWE has no effect on the other address bits besides the upper 4. For example, for an SoC using a 24-bit address bus, the upper 4 bits are ADDR[8:11]. See Address / Write Enable Multiplexing for additional details. 0 Default address function driven on 4 upper address bus signals 1 WE[0:3] driven into 4 upper address bus signals
22–23 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

Table continues on the next page...

EBI_CAL_ORn field descriptions (continued)

Field	Description
24–27 SCY	Cycle length in clocks This field represents the number of wait states (external cycles) inserted after the address phase in the single transfer case, or in the first beat of a burst, when the memory controller handles the external memory access . Values range from 0 to 15 . This is the main parameter for determining the length of the cycle. These bits are ignored when SETA=1. The total cycle length for the first beat (including the TS cycle) = (2+SCY) external clock cycles . See Example Wait State Calculation for related application information.
28 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
29–30 BSCY	Burst beats length in clocks This field determines the number of wait states (external cycles) inserted in all burst beats except the first, when the memory controller starts handling the external memory access and thus is using SCY[0:3] to determine the length of the first beat . These bits are ignored when SETA=1. The total memory access length for each beat is (1 + BSCY) external clock cycles. The total cycle length (including the TS cycle) = (2+SCY) + (#beats² - 1) * (BSCY+1).
31 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

2. #beats is the number of beats (4,8,16) determined by BL and PS bits in Base Register.

27.5 Functional Description

27.5.1 External Bus Interface Features

27.5.1.1 29-Bit Address Bus with Transfer Size Indication

The transfer size for an external transaction is indicated by the TSIZ[0:1] signals during the clock where address is valid. Valid transaction sizes are 8, 16 and 32 bits. SoCs may choose to pin-out less than 29 address pins externally (device-specific option), but a full 32-bit decode is done internally to determine the target of the transaction and whether a chip select should be asserted.

27.5.1.2 32-Bit Data Bus (16-bit Data Bus Mode also supported)

The entire 32-bit data bus is available for external memory accesses. There is also a 16-bit Data Bus Mode available via the DBM bit in EBI_MCR. See [16-Bit Data Bus Mode](#).

27.5.1.3 Multiplexed Address on Data Pins (singlemaster)

When this mode is enabled, the address shows up on the data pins during the address phase of the cycle. This mode can be enabled separately for non-chip-select accesses and per chip-select access. See [Multiplexed Address on Data Bus Mode](#).

27.5.1.4 Memory Controller with Support for Various Memory Types

The EBI contains a memory controller that supports a variety of memory types, including synchronous burst mode flash and SRAM, and asynchronous/legacy flash and SRAM with a compatible interface.

Each chip-select bank is configured via its own pair of Base and Option Registers. Each time an internal to external bus cycle access is requested, the internal address is compared with the base address of each valid Base Register (with 17 bits having mask). See the following figure. If a match is found, the attributes defined for this bank in its BR and OR are used to control the memory access. If a match is found in more than one bank, the lowest bank matched handles the memory access (e.g., bank 0 is selected over bank 1).

A match on a valid calibration chip-select register overrides a match on any non-calibration chip-select register, with CAL_CS0 having the highest priority. Thus the full priority of the chip-selects is: CAL_CS0,...,CAL_CS3,CS0,...,CS3

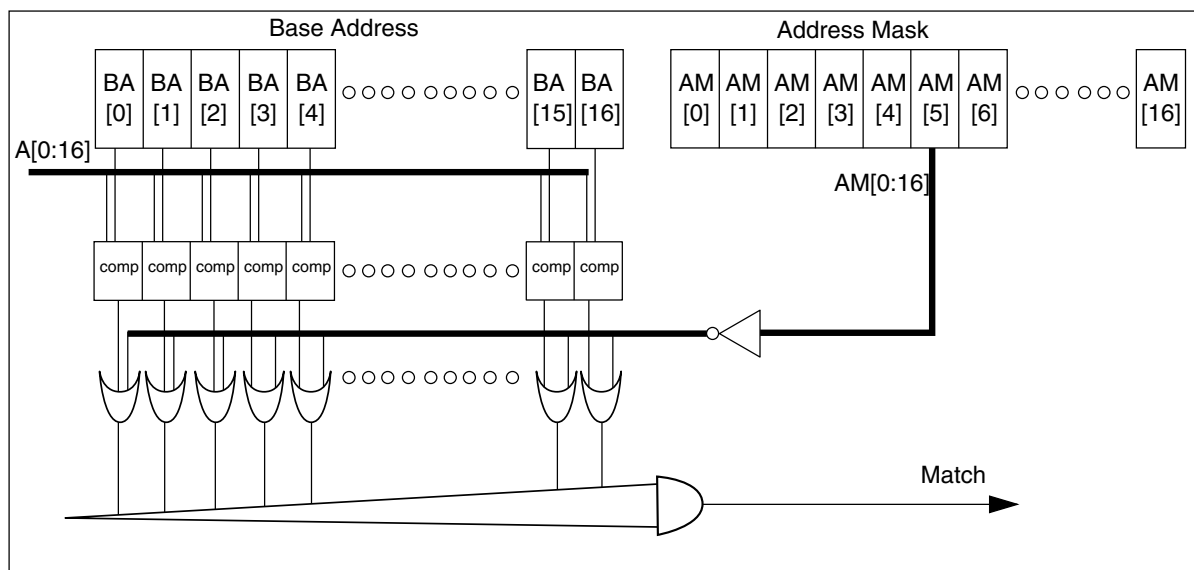


Figure 27-14. Bank Base Address & Match Structure

When a match is found on one of the chip-select banks, all its attributes (from the appropriate Base and Option Registers) are selected for the functional operation of the external memory access, such as:

- Number of wait states for a single memory access, and for any beat in a burst access
- Burst enable
- Port size for the external accessed device

See [/EBI Calibration Base Register \(EBI_CAL_BR \$n\$ \)](#) and [/EBI Calibration Option Register \(EBI_CAL_OR \$n\$ \)](#) for a full description of all chip-select attributes.

When no match is found on any of the chip-select banks, the default transfer attributes shown in the following table are used.

Table 27-35. Default Attributes for Non-Chip-Select Transfers

CS Attribute	Default Value	Comment
PS	0	32-bit port size
BL, SBL	0	burst length is don't care since burst is disabled
WEBS	0	$\overline{WE}$ not used for non-CS accesses
TBDIP	0	don't care since burst is disabled
BI	1	burst inhibited
SCY	0	don't care since external $\overline{TA}$ is used
BSCY	0	don't care since external $\overline{TA}$ is used
AD_MUX	0 ¹	Address on Data multiplexing
SETA	1	Select external $\overline{TA}$ to terminate access
EOE	0	$\overline{OE}$ not used for non-CS accesses
GCSN	0	$\overline{CS}/\overline{CAL\_CS}$ not used for non-CS accesses

1. For non-CS transfers, the AD_MUX attribute is determined by EBI_MCR[AD_MUX]

27.5.1.5 Burst Support (wrapped only)

The EBI supports burst read accesses of external burstable memory. To enable bursts to a particular memory region, clear the BI (Burst Inhibit) bit in the appropriate Base Register. External burst lengths of 2, 4 and 8 words are supported. Burst length is configured for each chip select by using the BL and SBL bits in the appropriate Base Register. See [Burst Transfer](#) for more details on burst operation.

In 16-bit data bus mode (DBM=1 in EBI_MCR), a special 2-beat burst case is supported for reads and writes for 32-bit non-chip-select accesses only. This is to allow 32-bit coherent accesses to another MCU. See [Non-Chip-Select Burst in 16-bit Data Bus Mode](#).

Bursting of accesses that are not controlled by the chip selects is not supported for any other case besides the special case of 32-bit accesses in 16-bit data bus mode.

Burst writes are not supported for any other case besides the special case of 32-bit non-chip-select writes in 16-bit data bus mode. Internal requests to write >32 bits (such as a cache line) externally are broken up into separate 32-bit or 16-bit external transactions according to the port size. See [Small Accesses \(Small Port Size and Short Burst Length\)](#) for more detail on these cases.

27.5.1.6 Bus Monitor

When enabled (via the BME bit in the EBI_BMCR), the bus monitor detects when no $\overline{TA}$ assertion is received within a maximum timeout period for external $\overline{TA}$ accesses. The timeout for the bus monitor is specified by the BMT field in the EBI_BMCR. Each time a timeout error occurs, the BMTF bit is set in the EBI_TESR. The timeout period is measured in external bus (CLKOUT) cycles. Thus the effective real-time period is multiplied (by 2, 3, etc.) when a slower-speed mode is used, even though the BMT field itself is unchanged.

27.5.1.7 Port Size Configuration per Chip Select (16 or 32 bits)

The EBI supports memories with data widths of 16 or 32 bits. The port size for a particular chip select is configured by writing the PS bit in the corresponding Base Register.

27.5.1.8 Configurable Wait States

From 0 to 15 wait states can be programmed for any cycle that the memory controller generates, via the SCY bits in the appropriate Option Register. From 0 to 3 wait states between burst beats can be programmed using the BSCY bits in the appropriate Option Register.

27.5.1.9 Configurable internal or external $\overline{TA}$ per chip select

Each chip select can be configured (via the SETA bit) to have $\overline{TA}$ driven internally (by the EBI), or externally (by an external device). See [/EBI Calibration Base Register \(EBI_CAL_BR \$n\$ \)](#) for more details on SETA bit usage.

27.5.1.10 Support for Dynamic Calibration with up to four chip selects

The EBI contains four calibration chip select signals, controlling four independent memory banks on an optional second external bus for calibration. See [Calibration Bus Operation](#) for more details on using the calibration bus.

27.5.1.11 Four Write/Byte Enable ($\overline{\text{WE}}/\overline{\text{BE}}$) Signals

The functionality of the $\overline{\text{WE}}[0:3]/\overline{\text{BE}}[0:3]$ signals depends on the value of the WEBS bit in the corresponding Base Register. Setting WEBS to 1 configures these pins as $\overline{\text{BE}}[0:3]$, while resetting it to 0 configures them as $\overline{\text{WE}}[0:3]$. $\overline{\text{WE}}[0:3]$ are asserted only during write accesses, while $\overline{\text{BE}}[0:3]$ is asserted for both read and write accesses. The timing of the $\overline{\text{WE}}[0:3]/\overline{\text{BE}}[0:3]$ signals remains the same in either case.

The upper Write/Byte Enable ($\overline{\text{WE}}0/\overline{\text{BE}}0$) indicates that the upper eight bits of the data bus (DATA[0:7]) contain valid data during a write/read cycle. The upper middle Write/Byte Enable ($\overline{\text{WE}}1/\overline{\text{BE}}1$) indicates that the upper middle eight bits of the data bus (DATA[8:15]) contain valid data during a write/read cycle. The lower middle Write/Byte Enable ($\overline{\text{WE}}2/\overline{\text{BE}}2$) indicates that the lower middle eight bits of the data bus (DATA[16:23]) contain valid data during a write/read cycle. The lower Write/Byte Enable ($\overline{\text{WE}}3/\overline{\text{BE}}3$) indicates that the lower eight bits of the data bus (DATA[24:31]) contain valid data during a write/read cycle.

Note

The exception to the preceding $\overline{\text{WE}}/\overline{\text{BE}}$ description is that for 16-bit port transfers (DBM=1 or PS=1), only the $\overline{\text{WE}}[0:1]/\overline{\text{BE}}[0:1]$ signals are used, regardless of whether DATA[0:15] or DATA[16:31] are selected (via the D16_31 bit in the EBI_MCR). This means for the case where DATA[16:31] are selected, that $\overline{\text{WE}}0$ indicates that DATA[16:23] contains valid data, and $\overline{\text{WE}}1$ indicates that DATA[24:31] contains valid data.

The Write/Byte Enable lines affected in a transaction for a 32-bit port (PS = 0) and a 16-bit port (PS=1) are shown in the following table. Only Big Endian byte ordering is supported by the EBI.

Table 27-36. Write/Byte Enable Signals Function 1

Transfer Size	TSIZ[0:1]	Address		32-Bit Port Size				Also applies when DBM=1 for 16-bit data bus mode. 16-Bit Port Size ²			
		A30	A31	WE0/BE0	WE1/BE1	WE2/BE2	WE3/BE3	WE0/BE0	WE1/BE1	WE2/BE2	WE3/BE3
Byte	01	0	0	X				X			
	01	0	1		X				X		
	01	1	0			X		X			
	01	1	1				X		X		
16-bit	10	0	0	X	X			X	X		
	10	1	0			X	X	X	X		
32-bit	00	0	0	X	X	X	X	X ^{3, 3}	X ³		
Burst	00	0	0	X	X	X	X	X	X		

1. This table applies to aligned internal master transfers only. In the case of a misaligned internal master transfer that is split into multiple aligned external transfers, not all of the write enables X'd in the table will necessarily assert. See [Misaligned Access Support](#).
2. Also applies when DBM=1 for 16-bit data bus mode.
3. This case consists of two 16-bit external transactions, but for both transactions the $\overline{WE}[0:1]/\overline{BE}[0:1]$ signals are the only $\overline{WE}/\overline{BE}$ signals affected.

27.5.1.12 Slower-Speed Clock Modes

For memories that cannot run with a full-speed external bus, the EBI supports slower-speed clock modes. See [Slower-Speed Modes](#) for more details on this feature. The timing diagrams for slower-speed modes are identical to those for full-speed mode, except that the frequency of CLKOUT is reduced.

27.5.1.13 Stop and Module Disable Modes for Power Savings

See [Modes of Operation](#) for a description of the power saving modes.

27.5.1.14 Optional Automatic CLKOUT Gating

The EBI has the ability to hold the external CLKOUT pin high when the EBI's internal master state machine is idle and no requests are pending. The EBI outputs a signal to the pads logic in the MCU to disable CLKOUT. This feature is disabled out of reset, and can be enabled or disabled by the ACGE bit in the EBI_MCR.

Note

This feature must be disabled for multi-master systems. In those cases, one master is getting its clock source from the other master and needs it to stay valid continuously.

27.5.1.15 Misaligned Access Support

The EBI has limited misaligned access support. Misaligned non-burst chip-select transfers from internal masters are supported. The EBI aligns the accesses when it sends them out to the external bus (splitting them into multiple aligned accesses if necessary), so that external devices are not required to support misaligned accesses. Burst accesses (internal master) must match the internal bus size (64-bit aligned). See [Misaligned Access Support](#) for more details.

27.5.1.16 Compatible with MPC5xx External Bus (with some limitations)

The EBI is compatible with the external bus of the MPC5xx parts, meaning that it supports most devices supported by the MPC5xx family of parts. However, there are some differences between this EBI and that of the MPC5xx parts that the user needs to be aware of before assuming that an MPC5xx-compatible device works with this EBI. See [Summary of Differences from MPC5xx](#) for details.

Note

Due to testing and complexity concerns, multi-master (or master/slave) operation between an eSys MCU and MPC5xx is not guaranteed.

27.5.2 External Bus Operations

The following sections provide a functional description of the external bus, the bus cycles provided for data transfer operations, and error conditions.

27.5.2.1 External Clocking

The CLKOUT signal sets the frequency of operation for the bus interface directly. Internally, the MCU uses a phase-locked loop (PLL) circuit to generate a master clock for all of the MCU circuitry (including the EBI) which is phase-locked to the CLKOUT

signal. In general, all signals for the EBI are specified with respect to the rising-edge of the CLKOUT signal, and they are guaranteed to be sampled as inputs or changed as outputs with respect to that edge.

27.5.2.2 Reset

Upon detection of internal reset assertion, the EBI immediately ends all transactions (abruptly, not through normal termination protocol), and ignores any transaction requests that take place while reset is asserted.

27.5.2.3 Basic Transfer Protocol

The basic transfer protocol defines the sequence of actions that must occur on the external bus to perform a complete bus transaction. A simplified scheme of the basic transfer protocol is shown in the following figure.

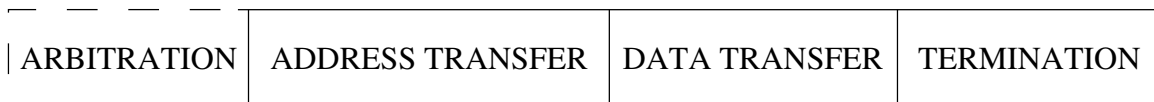


Figure 27-15. Basic Transfer Protocol

The arbitration phase is where bus ownership is requested and granted. This phase is not needed in Single Master Mode because the EBI is the permanent bus owner in this mode.

The address transfer phase specifies the address for the transaction and the transfer attributes that describe the transaction. The signals related to the address transfer phase are $\overline{TS}$, ADDR (or DATA if Address/Data multiplexing is used), $\overline{CAL\_CS}[0:3]$, $\overline{RD\_WR}$, TSIZ[0:1], and $\overline{BDIP}$. The address and its related signals (with the exception of $\overline{TS}$, $\overline{BDIP}$) are driven on the bus with the assertion of the $\overline{TS}$ signal, and kept valid until the bus master receives $\overline{TA}$ asserted (the EBI holds them one cycle beyond $\overline{TA}$ for writes and external $\overline{TA}$ accesses). Note that for writes with internal $\overline{TA}$, $\overline{RD\_WR}$ is not held one cycle past $\overline{TA}$.

The data transfer phase performs the transfer of data, from master to slave (in write cycles) or from slave to master (on read cycles), if any is to be transferred. The data phase may transfer a single beat of data (1-4 bytes) for non-burst operations or a 2-beat, 4-beat, 8-beat, or 16-beat burst of data (2 or 4 bytes per beat depending on Port Size) when burst is enabled. On a write cycle, the master must not drive write data until after the address transfer phase is complete. This is to avoid electrical contentions when switching between drivers. The master must start driving write data one cycle after the address transfer cycle. The master can stop driving the data bus as soon as it samples the $\overline{TA}$ line asserted on the rising edge of CLKOUT. To facilitate asynchronous write

support, the EBI keeps driving valid write data on the data bus until 1 clock after the rising edge where $\overline{\text{RD\_WR}}$ and $\overline{\text{WE}}$ are negated (for chip-select accesses only). See [Figure 27-22](#) for an example of write timing. On a read cycle, the master accepts the data bus contents as valid on the rising edge of the CLKOUT in which the $\overline{\text{TA}}$ signal is sampled asserted. See [Figure 27-17](#) for an example of read timing.

The termination phase is where the cycle is terminated by the assertion of either $\overline{\text{TA}}$ (normal termination) or $\overline{\text{TEA}}$ (termination with error). Termination is discussed in detail in [Termination Signals Protocol](#).

Note

In the timing diagrams in this document, asynchronous relationships between signals that switch in the same CLKOUT cycle are not guaranteed. For example, in [Figure 27-22](#), $\overline{\text{WE}}$ and write DATA change during the same CLKOUT cycle. There is no guarantee that DATA will be stable before $\overline{\text{WE}}$ assertion. External devices should not be latching write DATA on $\overline{\text{WE}}$ assertion, but instead must use a signal edge that takes place in a later CLKOUT cycle, such as $\overline{\text{WE}}$ negation.

Note

The timing diagrams in the following sections may show signals not implemented on some chips. Refer to [Table 27-5](#) for a list of your chip's signals.

27.5.2.4 Single Beat Transfer

The flow and timing diagrams in this section assume that the EBI is configured in Single Master Mode. Therefore, arbitration is not needed and is not shown in these diagrams.

27.5.2.4.1 Single Beat Read Flow

The handshakes for a single beat read cycle are illustrated in the following flow and timing diagrams.

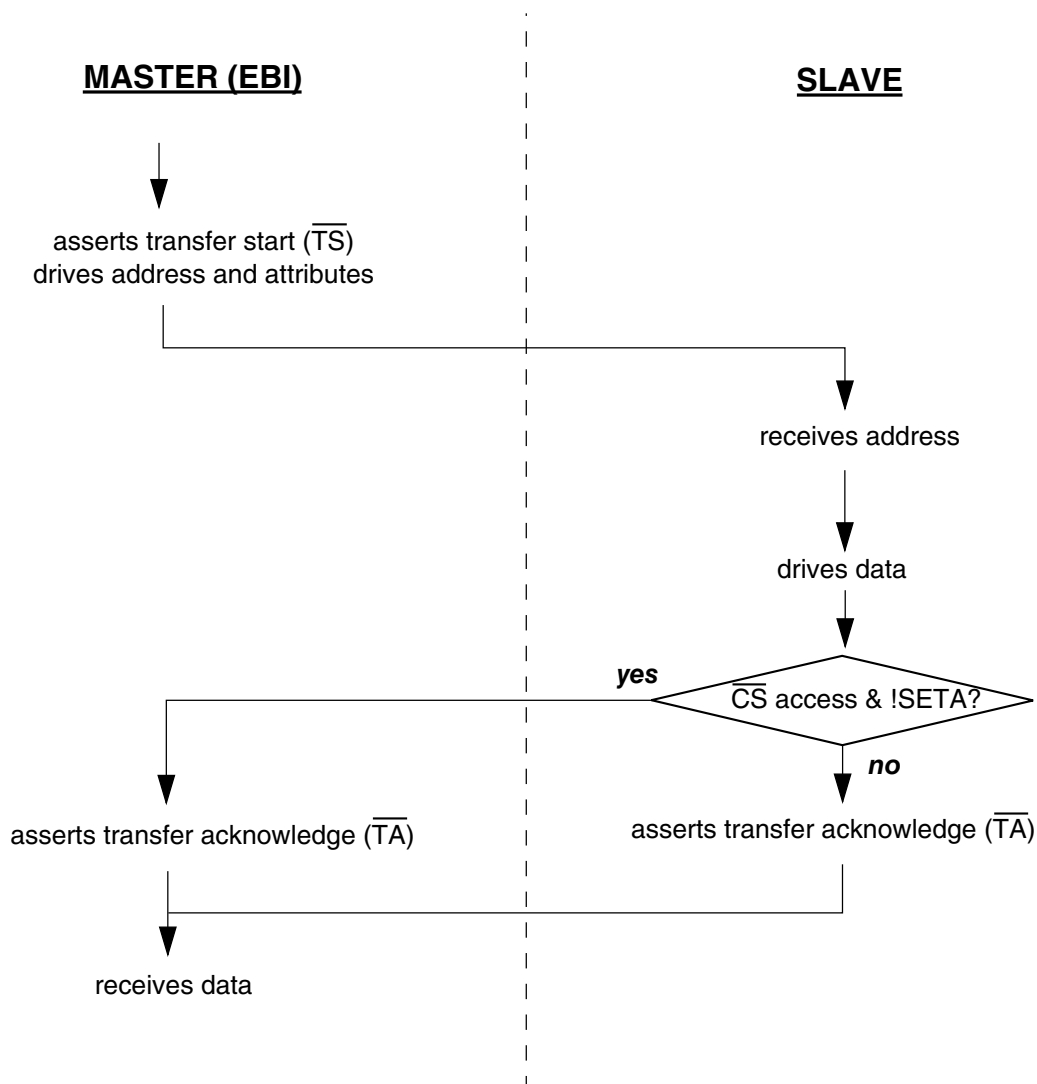


Figure 27-16. Basic Flow Diagram of a Single Beat Read Cycle

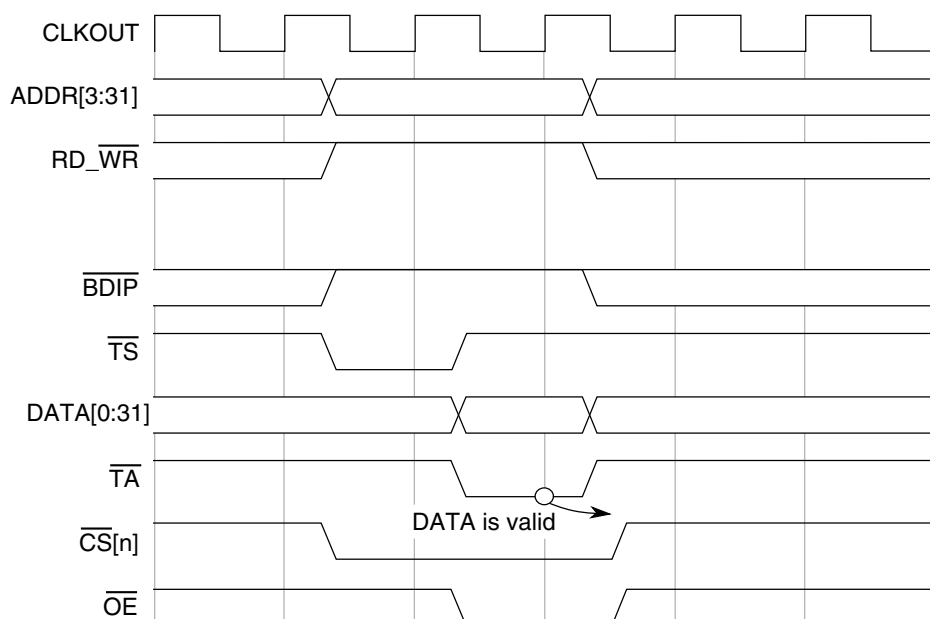


Figure 27-17. Single Beat 32-bit Read Cycle, $\overline{\text{CS}}$ Access, Zero Wait States

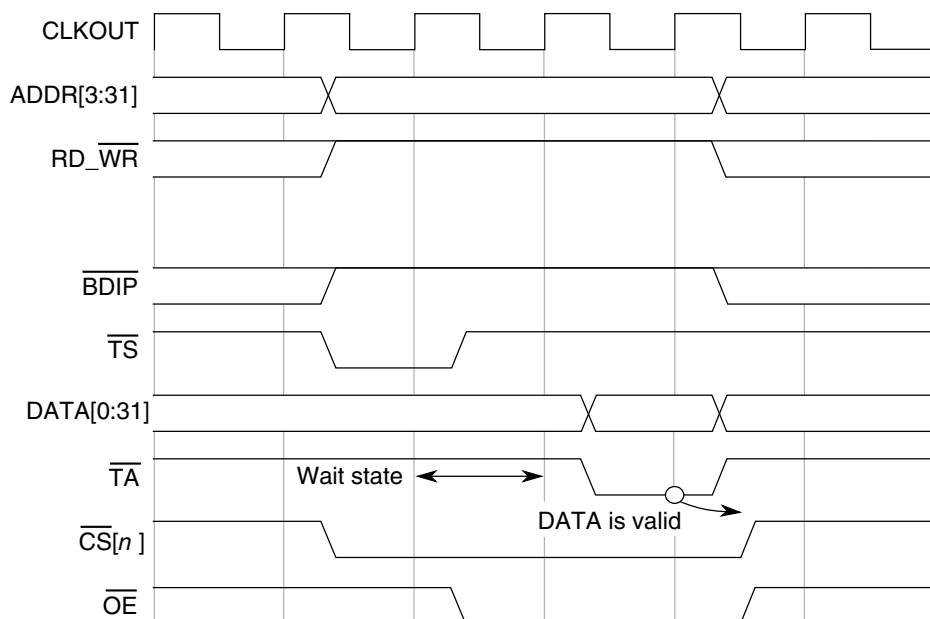
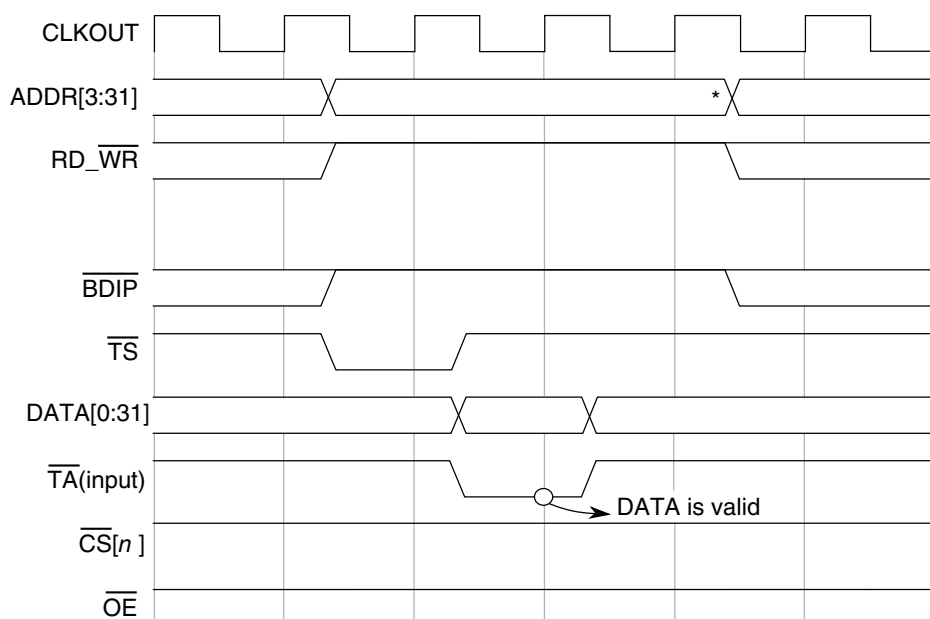


Figure 27-18. Single Beat 32-bit Read Cycle, $\overline{\text{CS}}$ Access, One Wait State



*The EBI drives address and control signals an extra cycle because it uses a latched version of the external $\overline{TA}$ (1 cycle delayed) to terminate the cycle.

Figure 27-19. Single Beat 32-bit Read Cycle, $\overline{CS}$ Access, SETA=1, Zero Wait States

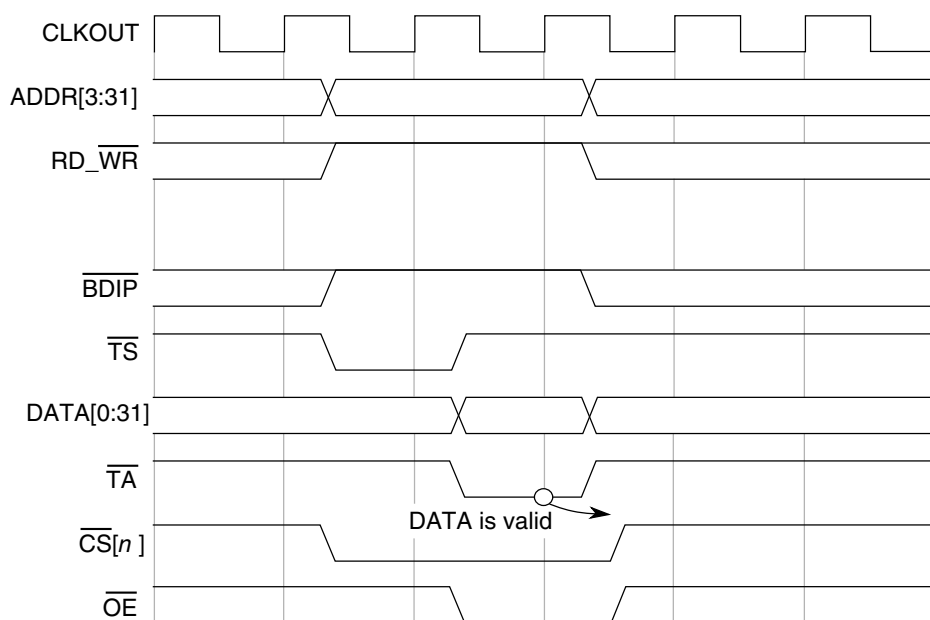
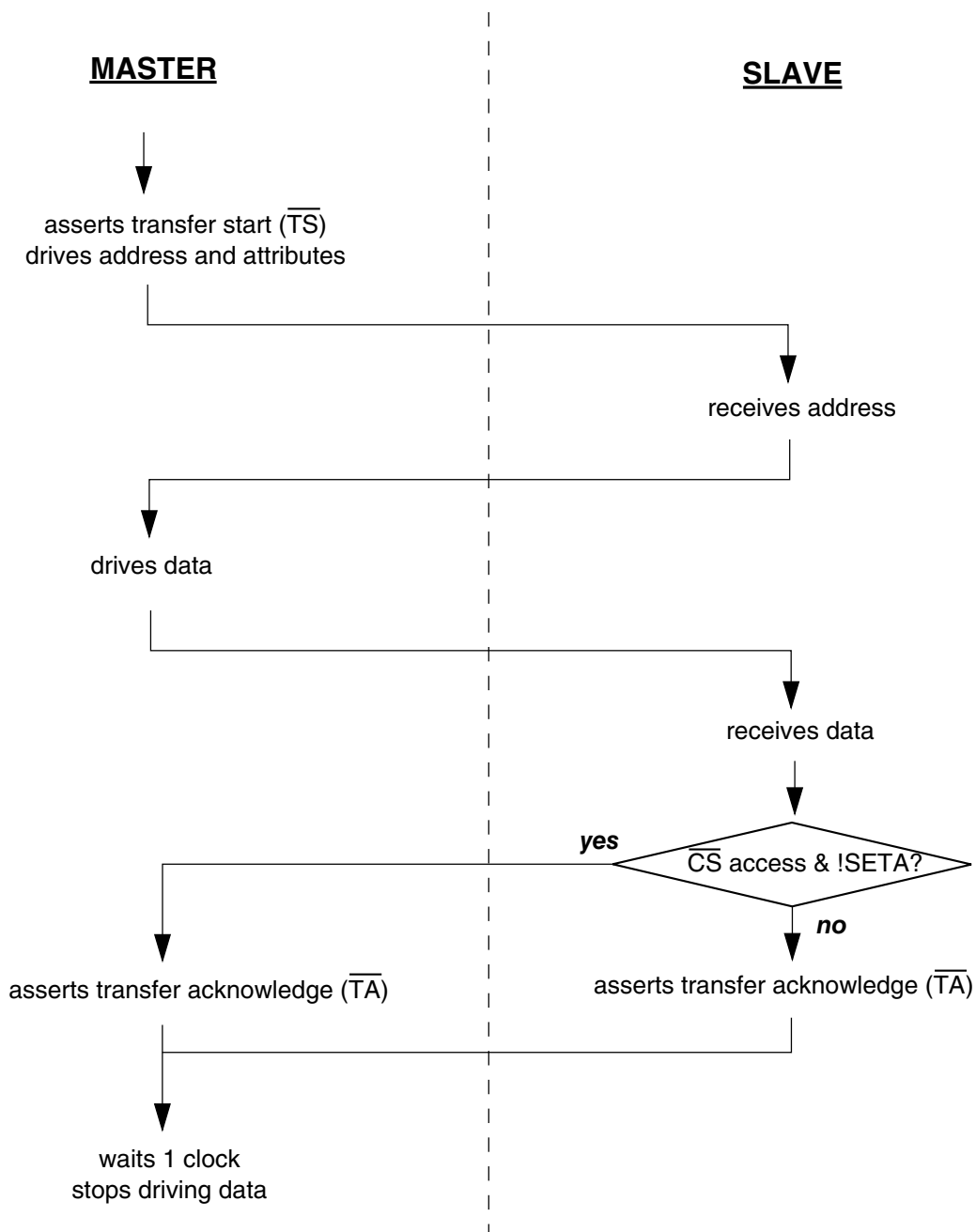


Figure 27-20. Single Beat 32-bit Read Cycle, $\overline{CS}$ Access, Zero Wait States (EOE=0b01, 0b10)

27.5.2.4.2 Single Beat Write Flow

The handshakes for a single beat write cycle are illustrated in the following flow and timing diagrams.

**Figure 27-21. Basic Flow Diagram of a Single Beat Write Cycle**

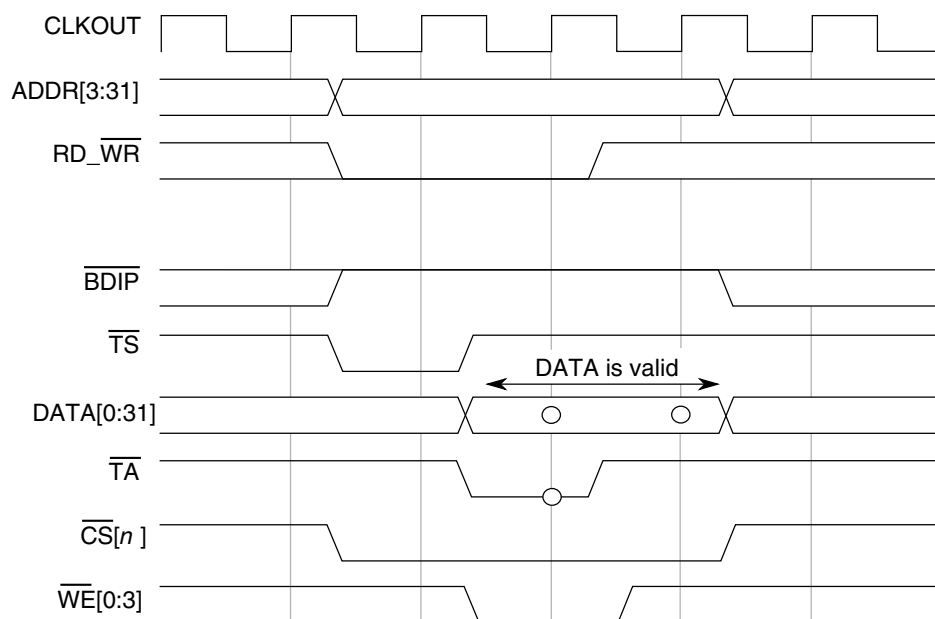


Figure 27-22. Single Beat 32-bit Write Cycle, $\overline{CS}$ Access, Zero Wait States

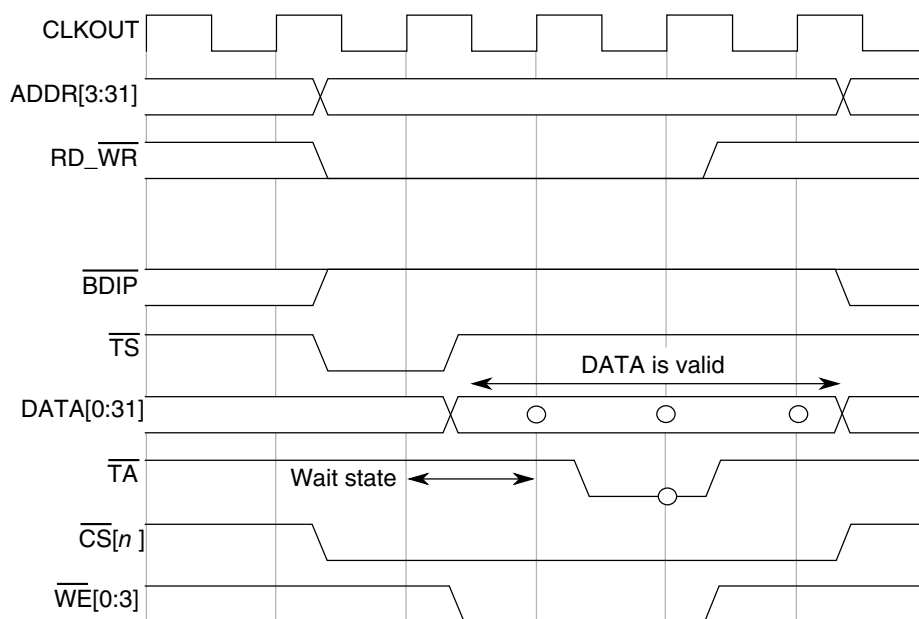
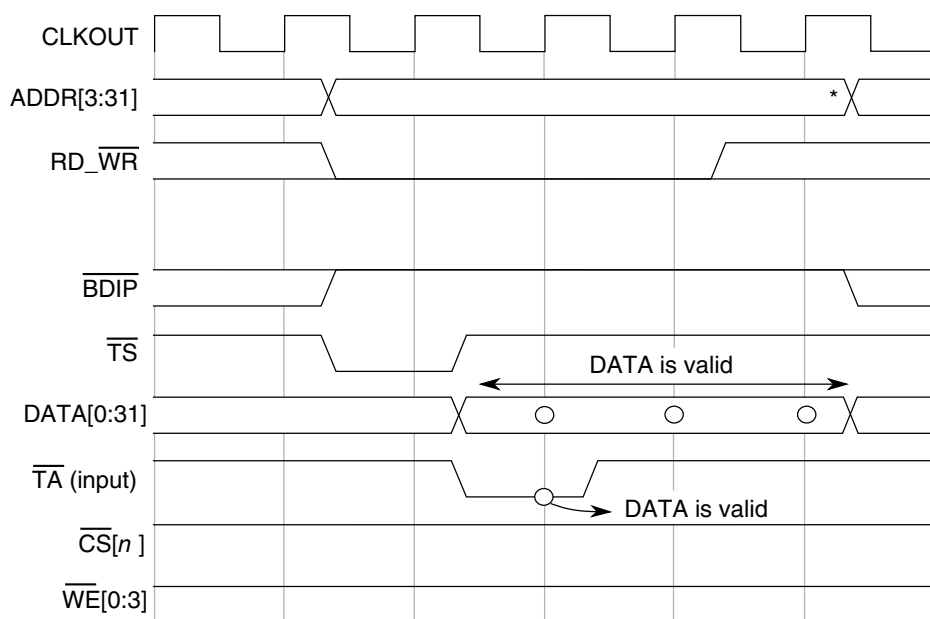
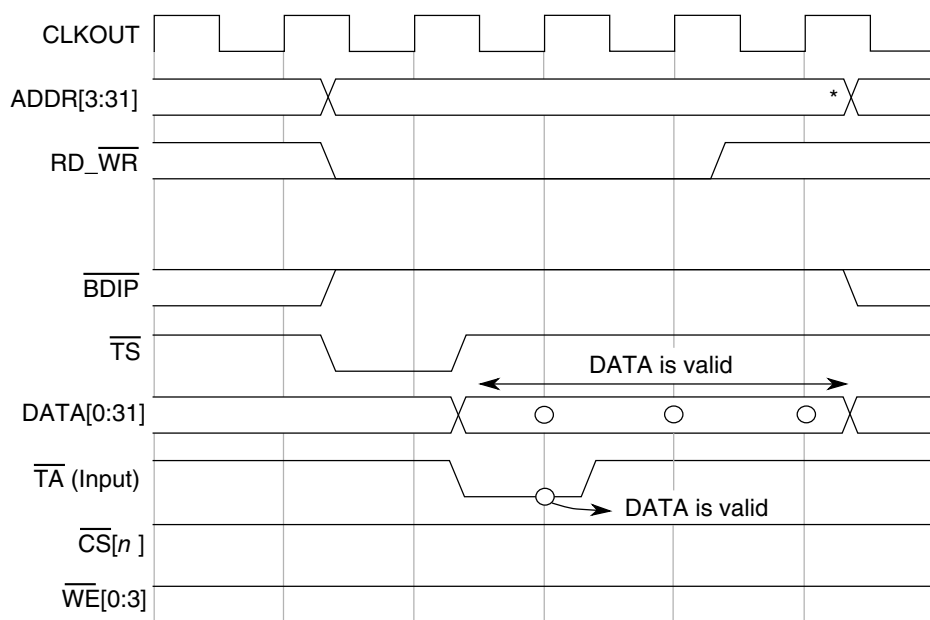


Figure 27-23. Single Beat 32-bit Write Cycle, $\overline{CS}$ Access, One Wait State



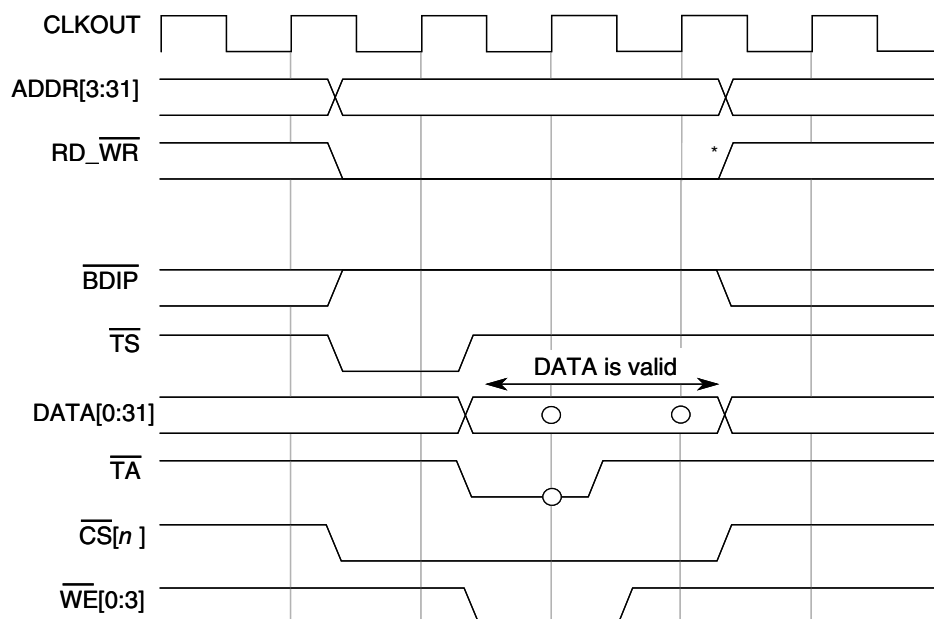
* The EBI drives address and control signals an extra cycle because it uses a latched version of the external $\overline{TA}$ (1 cycle delayed) to terminate the cycle.

Figure 27-24. Single Beat 32-bit Write Cycle, $\overline{CS}$ Access, SETA=1, Zero Wait States



* The EBI drives address and control signals an extra cycle because it uses a latched version of the external $\overline{TA}$ (1 cycle delayed) to terminate the cycle.

Figure 27-25. Single Beat 32-bit Write Cycle, Non- $\overline{CS}$ Access, Zero Wait States



* Negation of $\overline{RD\_WR}$ signal is in the same CLKOUT cycle as $\overline{CS}$ negation when $EBI\_BR[LWRN]=1$ (one cycle later than the default $LWRN=0$ timing).

Figure 27-26. Single Beat 32-bit Write Cycle, $\overline{CS}$ Access, Zero Wait States, LWRN=1

27.5.2.4.3 Back-to-Back Accesses

Due to internal bus protocol, one dead cycle is necessary between back-to-back external bus accesses that are not part of a set of small accesses (see [Small Accesses \(Small Port Size and Short Burst Length\)](#) for small access timing). A dead cycle refers to a cycle between the $\overline{TA}$ of a previous transfer and the $\overline{TS}$ of the next transfer.

Note

In some cases, $\overline{CS}$ remains asserted during this dead cycle, such as the cases of back-to-back writes or read-after-write to the same chip-select (with $EBI\_BR[GCSN]=0$). See [Figure 27-30](#) and [Figure 27-32](#). However, if $EBI\_BR[GCSN]=1$ for the first access (see [Figure 27-31](#) and [Figure 27-33](#)), then the EBI inserts an extra dead cycle between the accesses for these cases in order to guarantee $\overline{CS}$ negation between accesses.

Besides this dead cycle, in most cases, back-to-back accesses on the external bus do not cause any change in the timing from that shown in the previous diagrams, and the two transactions are independent of each other. The only exceptions to this are listed below:

- Back-to-back accesses where the first access ends with an externally-driven $\overline{\text{TA}}$ or $\overline{\text{TEA}}$. In these cases, an extra cycle is required between the end of the first access and the $\overline{\text{TS}}$ assertion of the second access. See [Termination Signals Protocol](#) for more details.
- 4-word burst transfers (using 64-bit AMBA bus) whose starting address is not 0x20 aligned. In these cases, an extra cycle is required between the end of the 2nd 4-word burst access and the $\overline{\text{TS}}$ assertion of a subsequent access. See [Figure 27-44](#).

The following diagrams show a few examples of back-to-back accesses on the external bus.

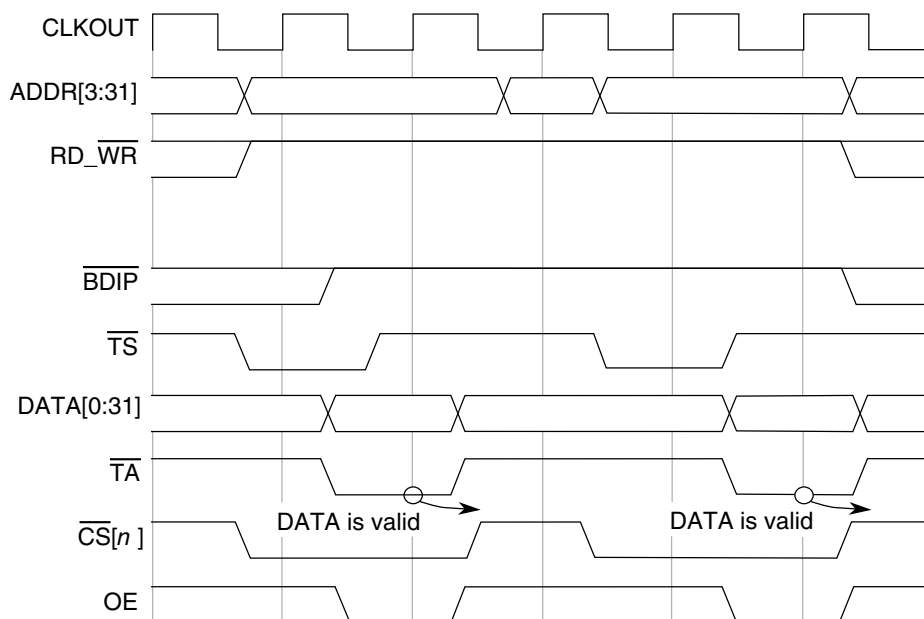


Figure 27-27. Back-to-Back 32-bit Reads to the Same $\overline{\text{CS}}$ Bank

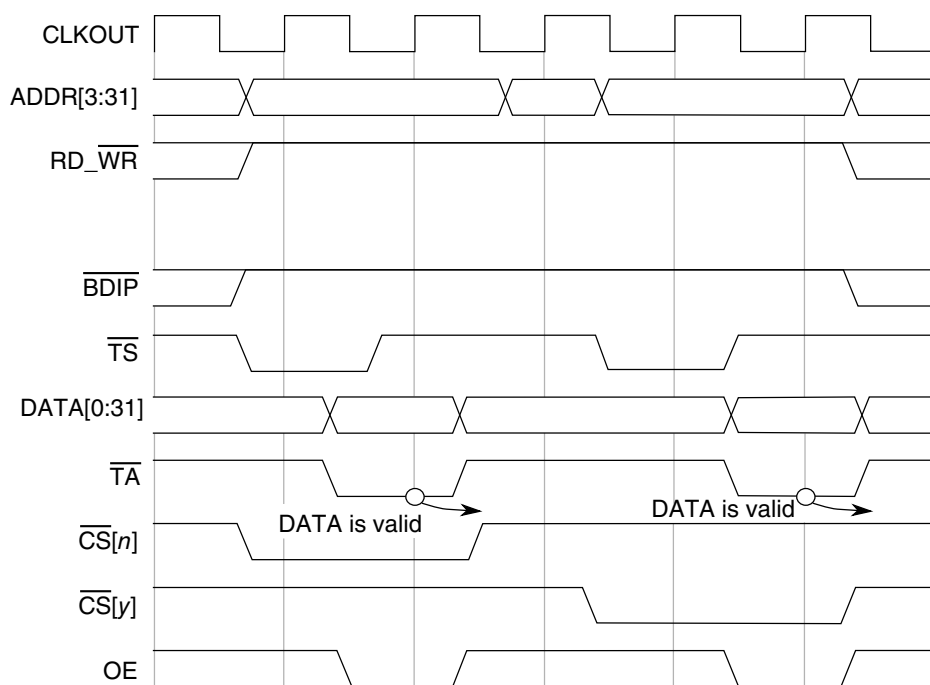


Figure 27-28. Back-to-Back 32-bit Reads to Different $\overline{CS}$ Banks

Functional Description

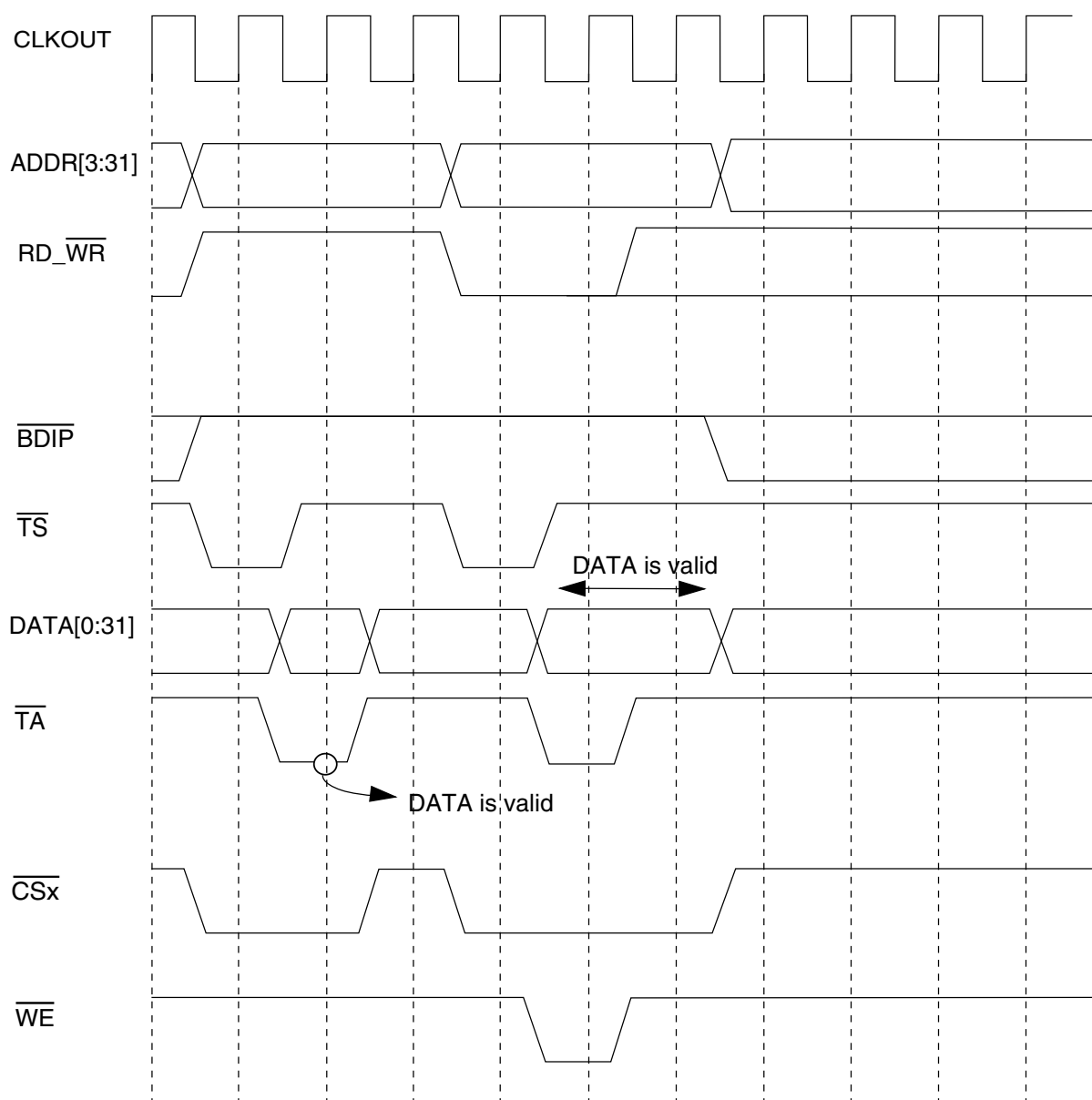


Figure 27-29. Write After Read to the Same $\overline{\text{CS}}$ Bank

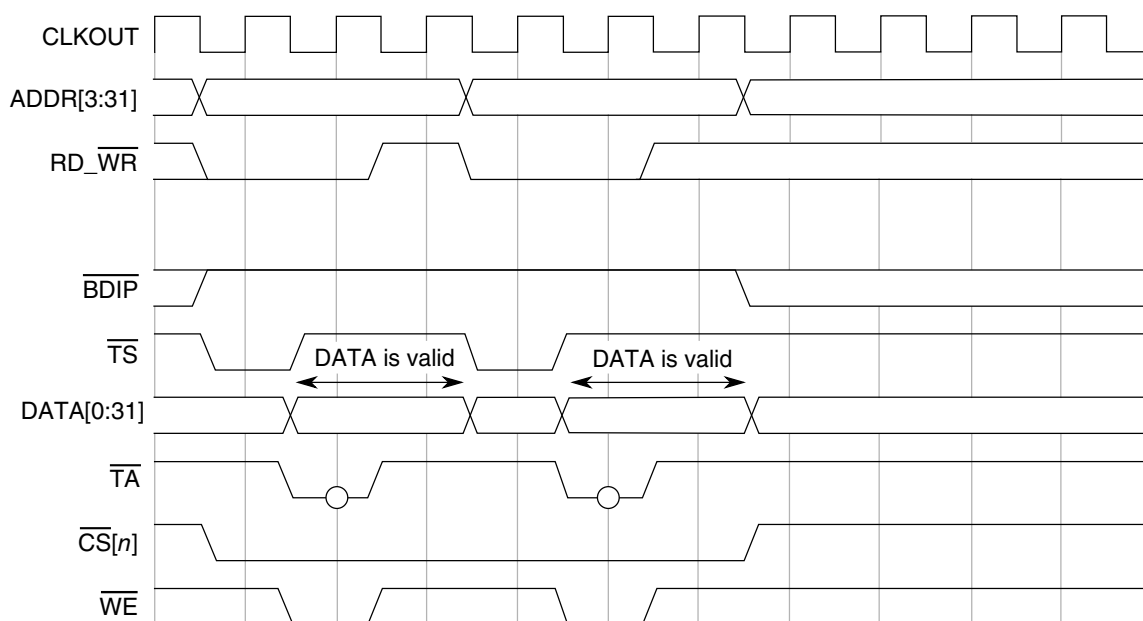


Figure 27-30. Back-to-Back 32-bit Writes to the Same $\overline{CS}$ Bank, $GCSN=0$

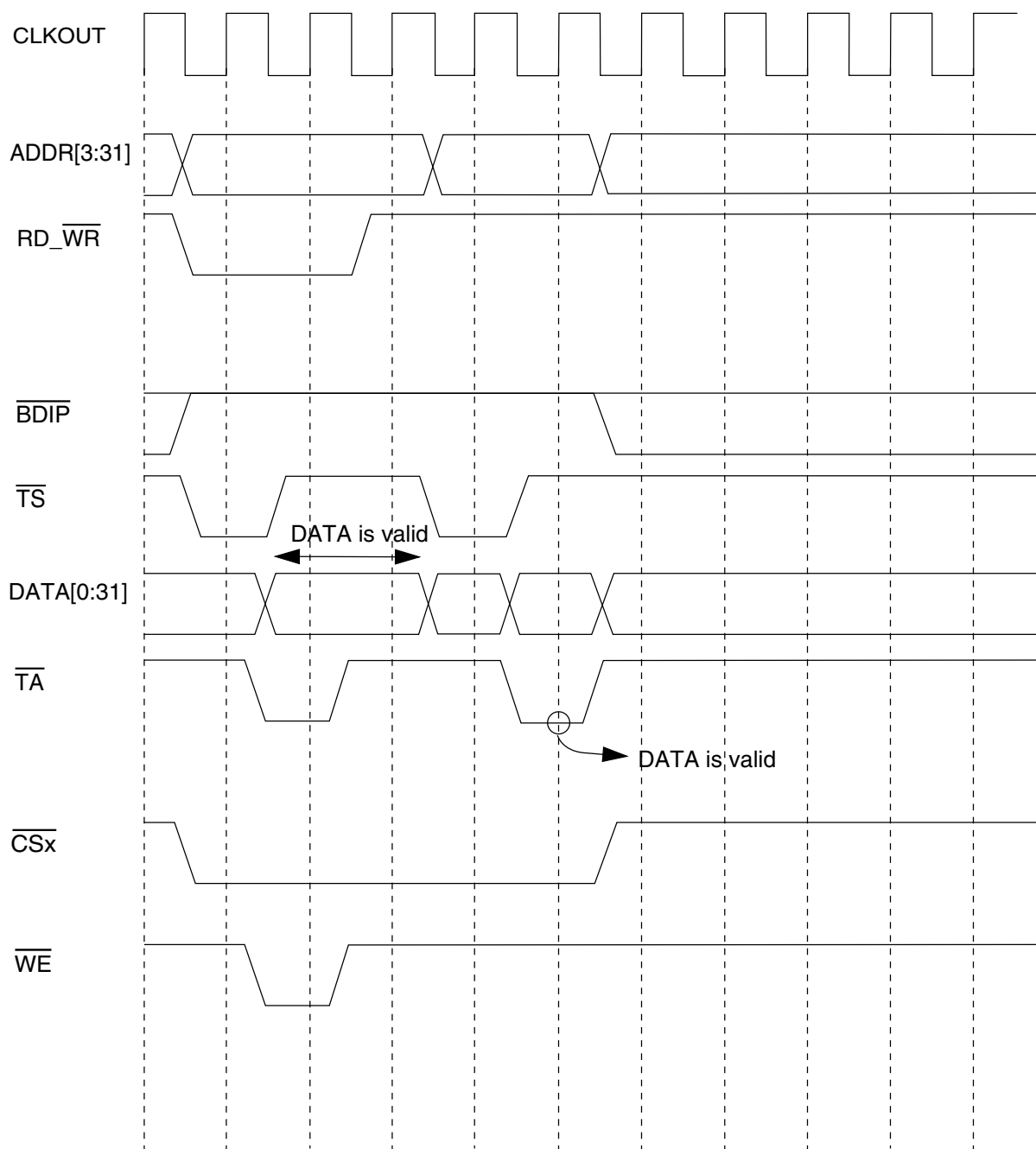


Figure 27-31. Back-to-Back 32-bit Writes to the Same $\overline{CS}$ Bank, $GCSN=1$

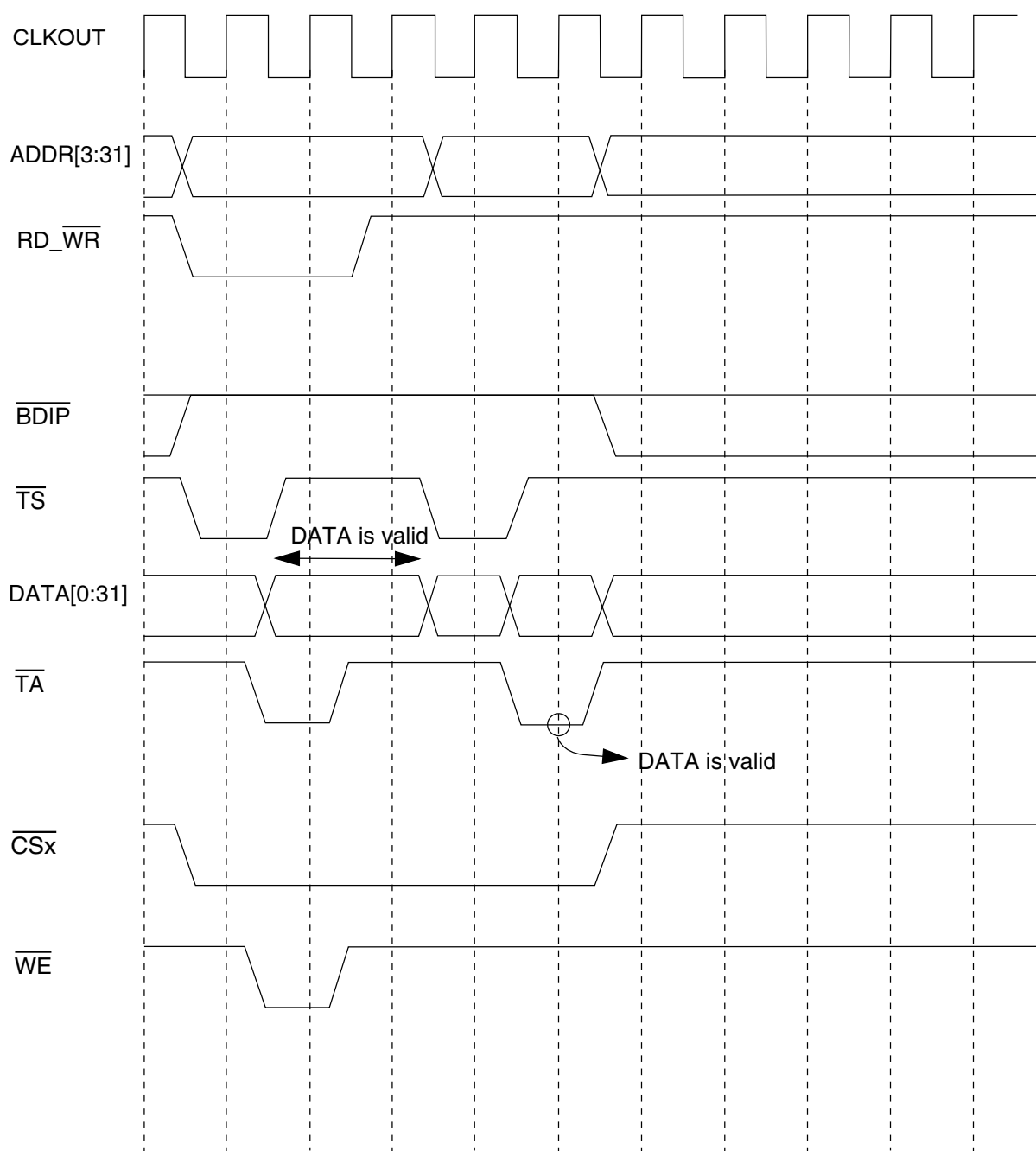


Figure 27-32. Read After Write to the Same $\overline{CS}$ Bank, $GCSN=0$

Functional Description

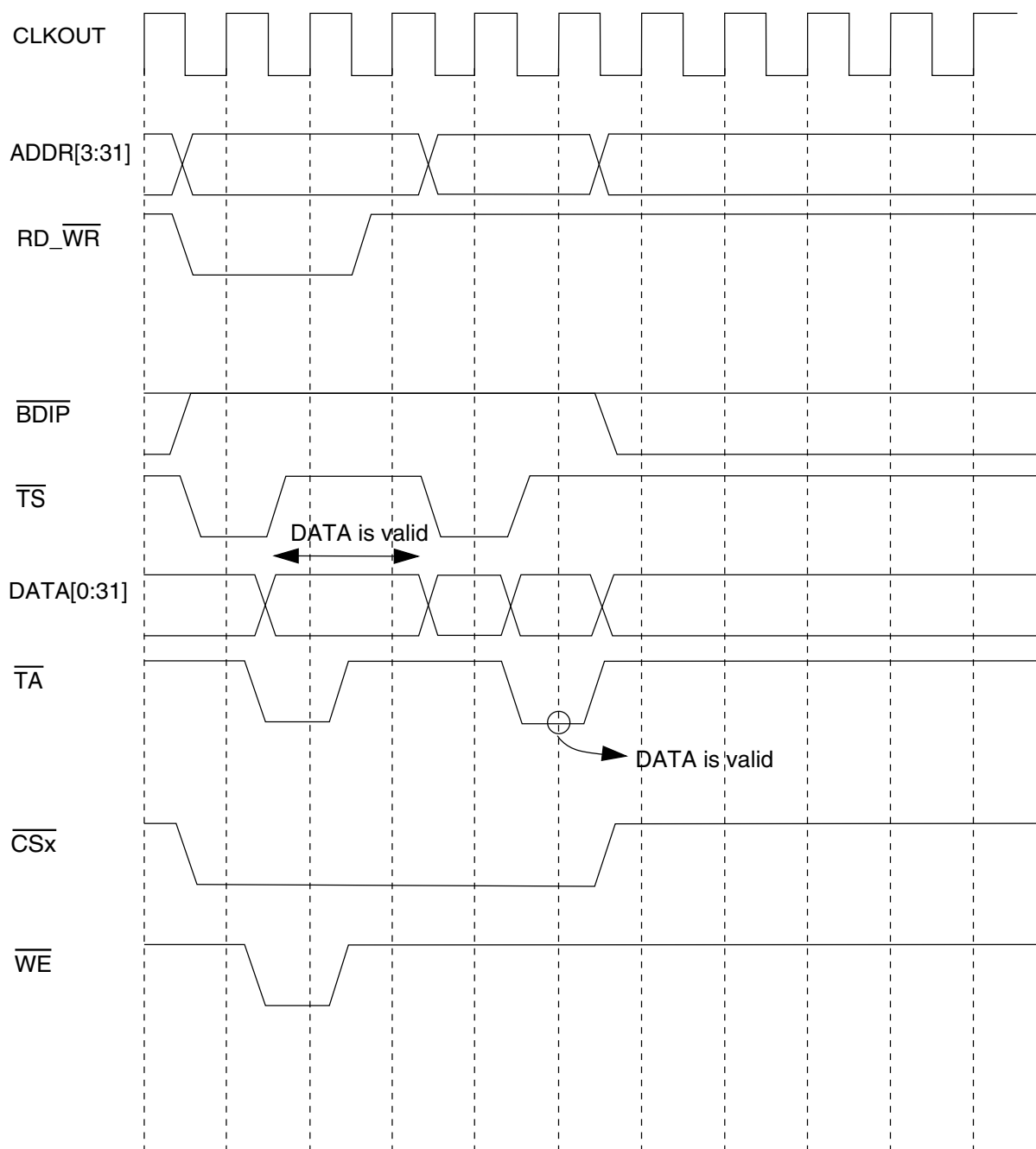


Figure 27-33. Read After Write to the Same $\overline{\text{CS}}$ Bank, GCSN=1

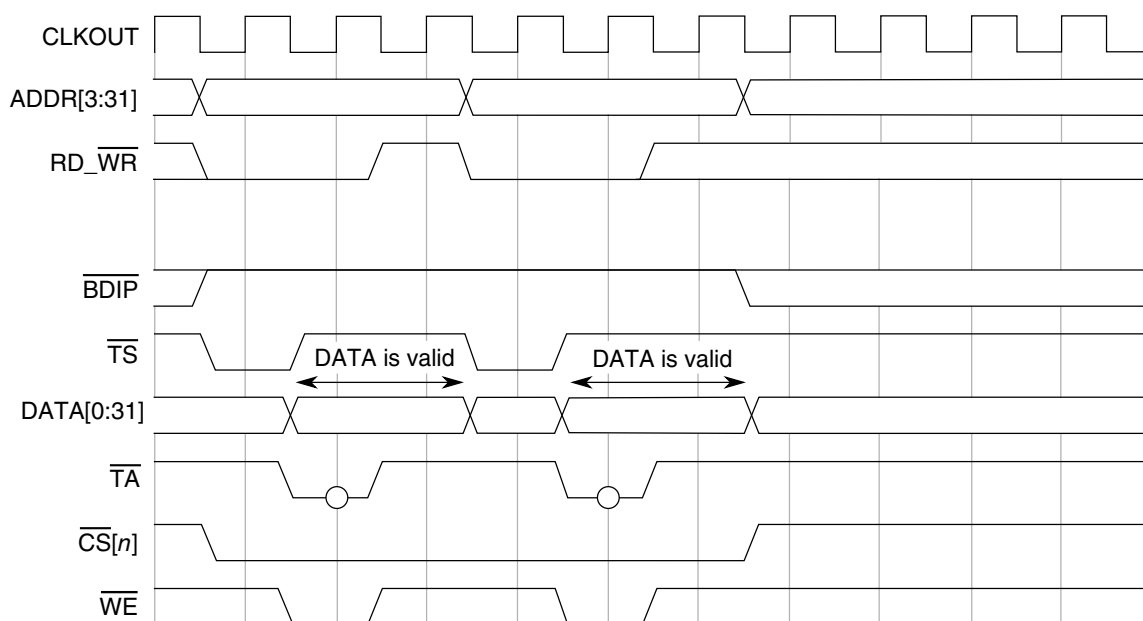


Figure 27-34. Back-to-Back 32-bit Writes to Different $\overline{CS}$ Banks (GCSN=0)

Functional Description

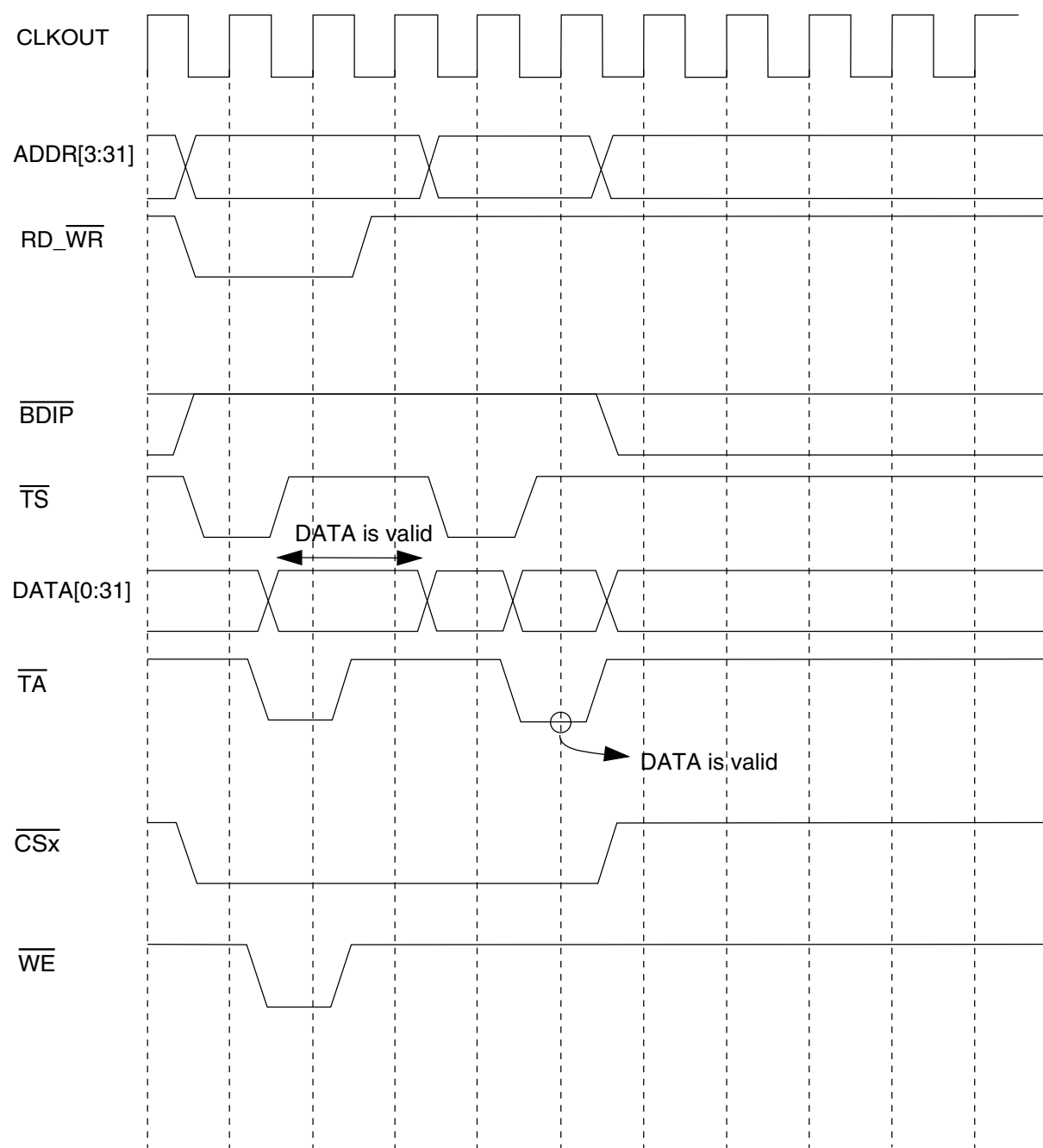


Figure 27-35. Back-to-Back 32-bit Writes to the Same $\overline{CS}$ Bank, GCSN=1

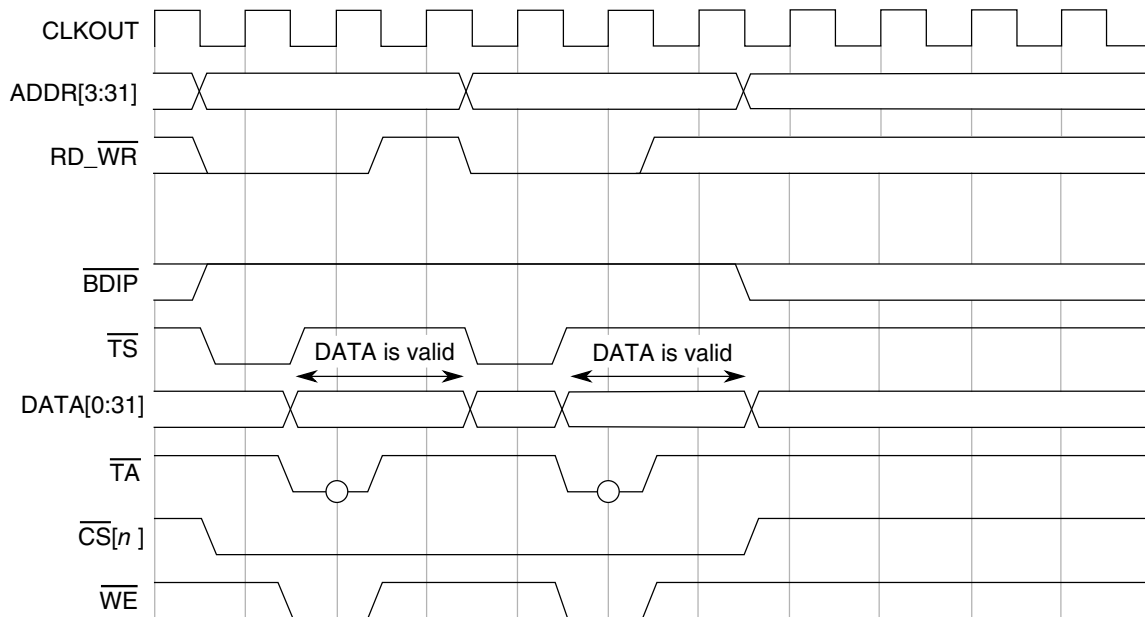


Figure 27-36. Read After Write to Different $\overline{CS}$ Banks

27.5.2.5 Burst Transfer

The EBI supports wrapping 32-byte or 16-byte critical-doubleword-first burst transfers. Bursting is supported only for internally-requested (e.g. core, dma, etc.) 32-byte or 16-byte read accesses to external devices that use the chip selects⁶.

Accesses to devices operating without a chip select are always single beat. If an internal request to the EBI indicates a size of less than 32 bytes, the request is fulfilled by running one or more single-beat external transfers, not by an external burst transfer.

An 8-word wrapping burst reads eight 32-bit words by supplying a starting address that points to one of the words (doubleword aligned) and requiring the memory device to sequentially drive each word on the data bus. The selected slave device must internally increment ADDR[27:29] (also ADDR30 in the case of a 16-bit port size device) of the supplied address for each transfer, until the address reaches an 8-word boundary, and then wrap the address to the beginning of the 8-word boundary. The address and transfer attributes supplied by the EBI remain stable during the transfers. Termination of each beat transfer occurs by the EBI asserting $\overline{TA}$ (SETA=1 is not supported for burst transfers). The EBI requires that addresses be aligned to a doubleword boundary on all burst cycles.

The following table shows the burst order of beats returned for an 8-word burst to a 32-bit port.

6. Except for the special case of a 32-bit non-chip-select access in 16-bit data bus mode. See [Non-Chip-Select Burst in 16-bit Data Bus Mode](#).

Table 27-37. Wrap Bursts Order

Burst Starting Address ADDR[27:28]	Burst Order(Assuming 32-bit Port Size)
00	word0 -> word1 -> word2 -> word3 -> word4 -> word5 -> word6 -> word7
01	word2 -> word3 -> word4 -> word5 -> word6 -> word7 -> word0 -> word1
10	word4 -> word5 -> word6 -> word7 -> word0 -> word1 -> word2 -> word3
11	word6 -> word7 -> word0 -> word1 -> word2 -> word3 -> word4 -> word5

The general case of burst transfers assumes that the external memory has 32-bit port size and 8-word burst length (SBL=0, BL=0). The EBI can also burst from 16-bit port size memories, taking twice as many external beats to fetch the data as compared to a 32-bit port with the same burst length. The EBI can also burst from 16-bit or 32-bit memories that have a 4-word burst length (SBL=0, BL=1 in the appropriate Base Register). In this case, two external 4-word burst transfers (wrapping on 4-word boundary) are performed to fulfill the internal 8-word request⁷. This operation is considered atomic by the EBI, so the EBI does not allow other unrelated master accesses or bus arbitration to intervene between the transfers. For more details and a timing diagram, see [Small Access Example #3: 32-byte Read to 32-bit Port with BL=1](#).

The EBI can also burst from 16-bit or 32-bit memories that have a 2-word burst length (SBL=1, BL=X in the appropriate Base Register). In this case, four⁸ external 2-word burst transfers (wrapping on 2-word boundary) are performed to fulfill the internal 8-word request. This operation is considered atomic by the EBI, so the EBI does not allow other unrelated master accesses or bus arbitration to intervene between the transfers. For more details and a timing diagram, see [Small Access Example #5: 32-byte Read to 32-bit Port with SBL=1](#).

During burst cycles, the $\overline{\text{BDIP}}$ (Burst Data In Progress) signal is used to indicate the duration of the burst data. During the data phase of a burst read cycle, the EBI receives data from the addressed slave. If the EBI needs more than one data, it asserts the $\overline{\text{BDIP}}$ signal. Upon receiving the data prior to the last data, the EBI negates $\overline{\text{BDIP}}$. Thus, the slave stops driving new data after it receives the negation of $\overline{\text{BDIP}}$ on the rising edge of the clock. Some slave devices have their burst length and timing configurable internally and thus may not support connecting to a $\overline{\text{BDIP}}$ pin. In this case, $\overline{\text{BDIP}}$ is driven by the EBI normally, but the output is ignored by the memory and the burst data behavior is determined by the internal configuration of the EBI and slave device. When the TBDIP bit is set in the appropriate Base Register, the timing for $\overline{\text{BDIP}}$ is altered. See [TBDIP Effect on Burst Transfer](#) for this timing.

7. This case (of 2 external 4-word burst transfers being required) applies only to AMBA data bus width of 64 bits.

8. Applies to 64-bit AMBA data bus width. For 32-bit AMBA data bus width, only two transfers are performed.

Since burst writes are not supported by the EBI⁹, the EBI negates $\overline{\text{BDIP}}$ during write cycles.

9. Except for the special case of a 32-bit non-chip-select access in 16-bit data bus mode. See [Non-Chip-Select Burst in 16-bit Data Bus Mode](#).

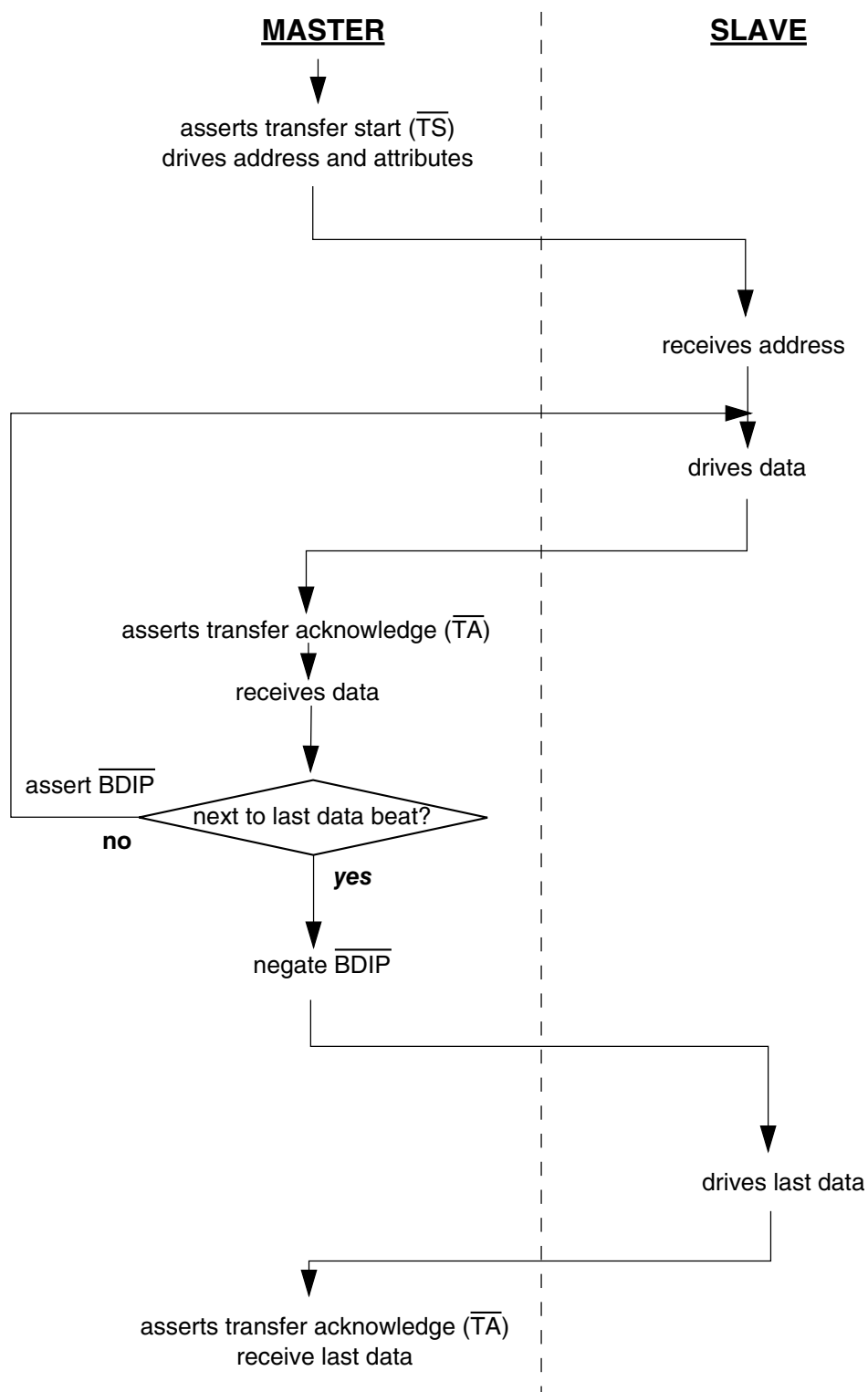


Figure 27-37. Basic Flow Diagram of a Burst Read Cycle

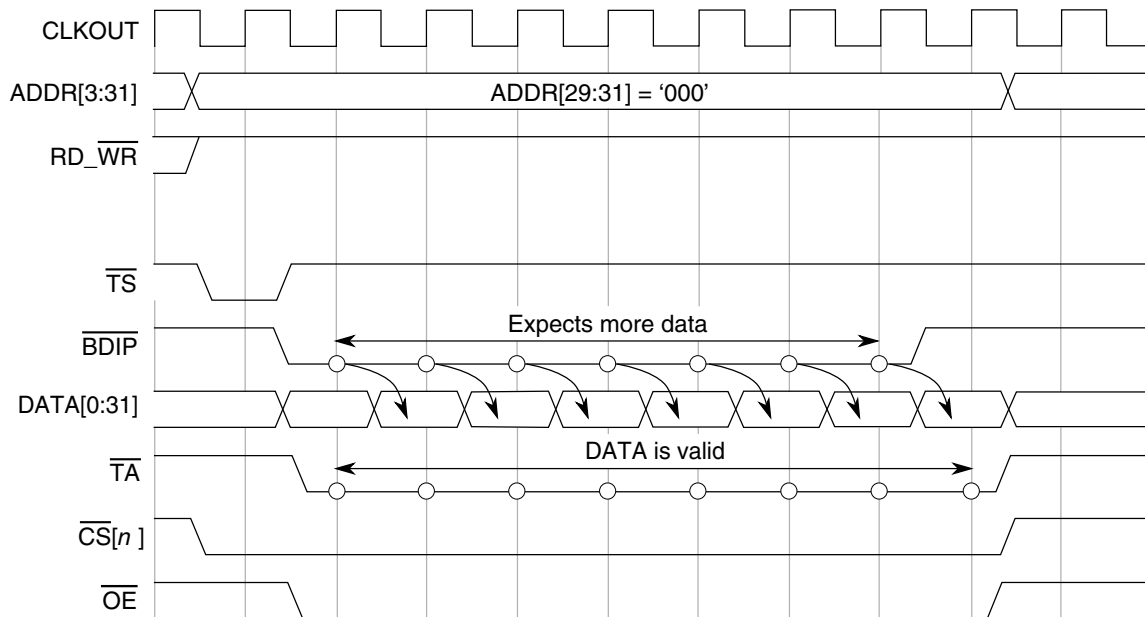


Figure 27-38. Burst 32-bit Read Cycle, Zero Wait States

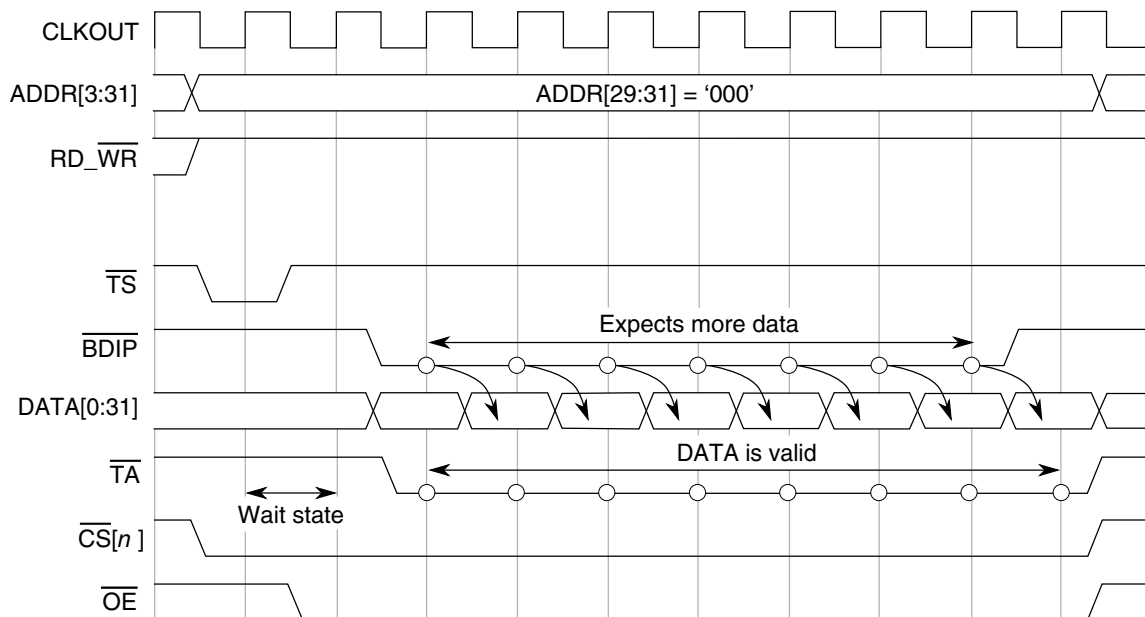


Figure 27-39. Burst 32-bit Read Cycle, One Initial Wait State

27.5.2.5.1 TBDIP Effect on Burst Transfer

Some memories require different timing on the $\overline{\text{BDIP}}$ signal than the default to run burst cycles. Using the default value of $\text{TBDIP}=0$ in the appropriate EBI Base Register results in $\overline{\text{BDIP}}$ being asserted ($\text{SCY}+1$) cycles after the address transfer phase, and being held asserted throughout the cycle regardless of the wait states between beats (BSCY). The following figure shows an example of the $\text{TBDIP}=0$ timing for a 4-beat burst with $\text{BSCY}=1$.

Functional Description

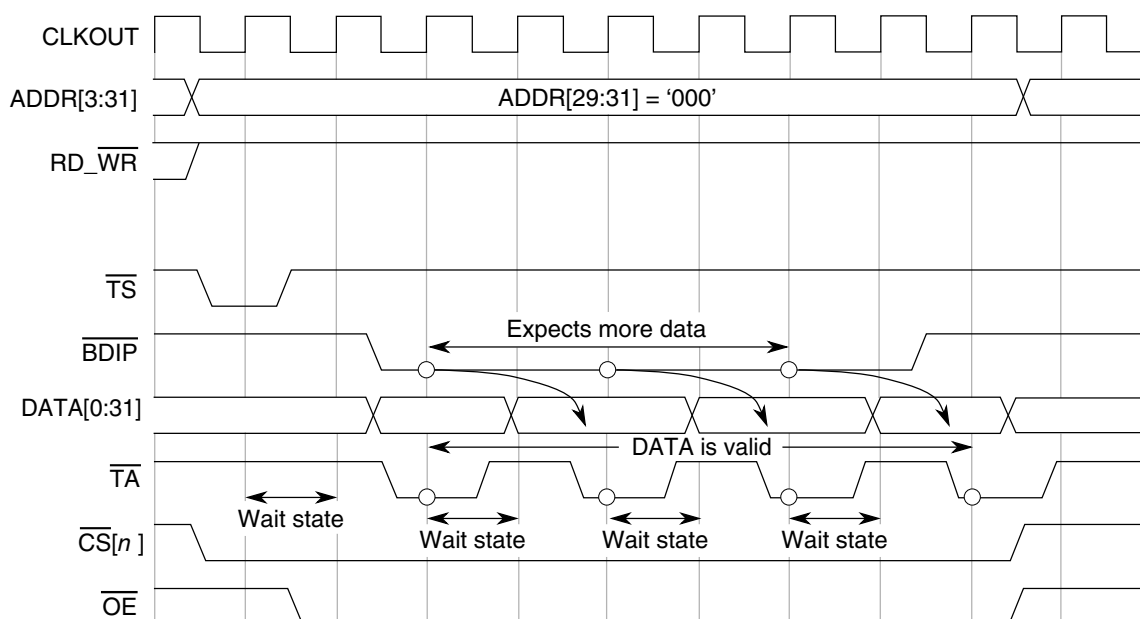


Figure 27-40. Burst 32-bit Read Cycle, One Wait State between Beats, TBDIP=0

When using TBDIP=1, the $\overline{\text{BDIP}}$ behavior changes to toggle between every beat when BSCY is a non-zero value. [Figure 27-41](#) shows an example of the TBDIP=1 timing for the same 4-beat burst shown in [Figure 27-40](#).

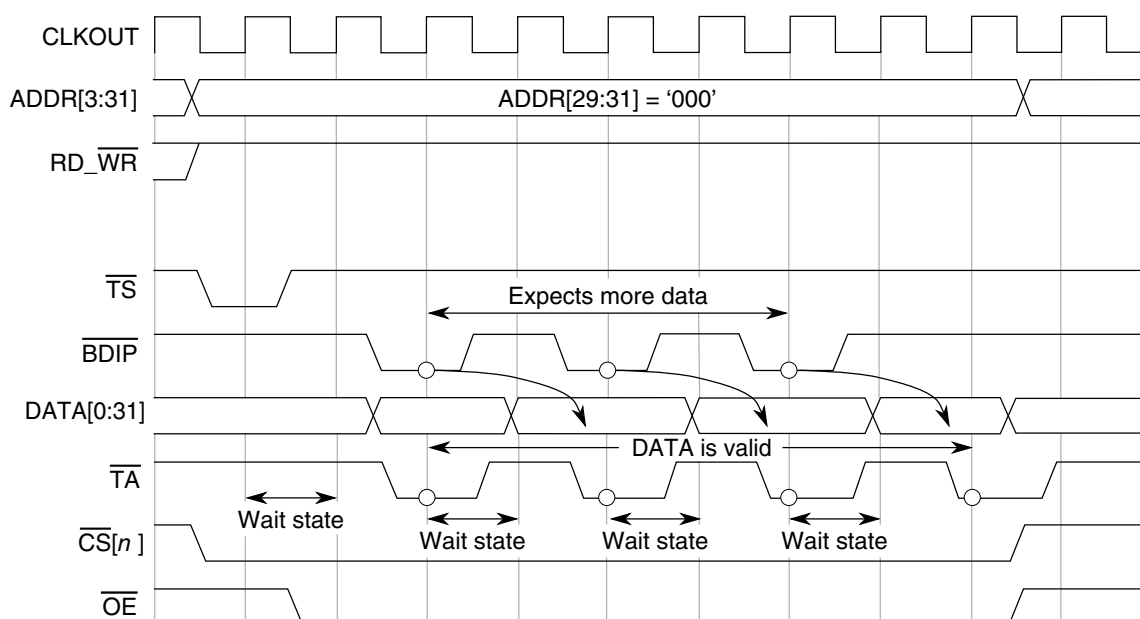


Figure 27-41. Burst 32-bit Read Cycle, One Wait State between Beats, TBDIP=1

27.5.2.6 Small Accesses (Small Port Size and Short Burst Length)

In this context, a small access refers to an access whose burst length and port size (SBL, BL, PS bits in Base Register for chip-select access or default burst disabled, 32-bit port for non-chip-select access) are such that the number of bytes requested by the internal master cannot all be fetched (or written) in one external transaction. If this is the case, the EBI initiates multiple transactions until all the requested data is transferred. It should be noted that all the transactions initiated to complete the data transfer are considered as an atomic transaction, so the EBI does not allow other unrelated master accesses to intervene between the transfers.

The following table shows all the combinations of burst length, port size, and requested byte count that cause the EBI to run multiple external transactions to fulfill the request.

Table 27-38. Small Access Cases

Byte Count Requested by internal master	Burst Length	Port Size	# External Accesses to Fulfill Request
Non-Burstable Chip-Select Banks (BI=1) or Non-Chip-Select Access			
4	1 beat	16-bit	2/1 ¹
8	1 beat	32-bit	2
8	1 beat	16-bit	4
16 ^{2, 2}	1 beat	32-bit	4
16 ²	1 beat	16-bit	8
32 ^{3, 3}	1 beat	32-bit	8
32 ³	1 beat	16-bit	16
Burstable Chip-Select Banks (BI=0)			
32 ³	4 words	16-bit (8 beats), 32-bit (4 beats)	2
32 ⁴	2 words	16-bit (4 beats), 32-bit (2 beats)	4

1. In 32-bit data bus mode (DBM=0 in EBI_MCR), two accesses are performed. In 16-bit data bus mode (DBM=1), one 2-beat burst access is performed and this is not considered a "small access" case. See [Non-Chip-Select Burst in 16-bit Data Bus Mode](#) for this special DBM=1 case.
2. Only supported for case of 32-bit internal AMBA data bus.
3. Only supported for case of 64-bit internal AMBA data bus.
4. SBL=1 (2-word burst case).

In most cases, the timing for small accesses is the same as for normal single-beat and burst accesses, except that multiple back-to-back external transfers are executed for each internal request. These transfers have no additional dead cycles in-between that are not present for back-to-back stand-alone transfers except for the case of writes with an internal request size of >64 bits, discussed in [Small Access Example #2: 32-byte Write with External \$\overline{TA}\$](#) .

The following sections show a few examples of small accesses. The timing for the remaining cases in the preceding table can be extrapolated from these and the other timing diagrams in this document.

27.5.2.6.1 Small Access Example #1: 32-bit Write to 16-bit Port

The following figure shows an example of a 32-bit write to a 16-bit port, requiring two 16-bit external transactions.

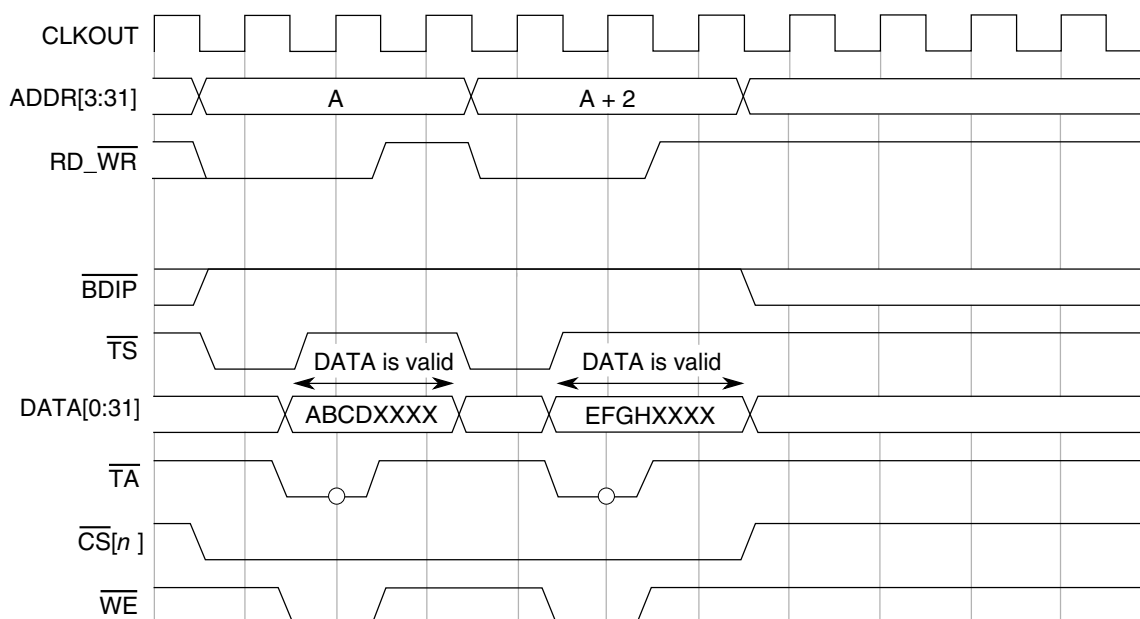
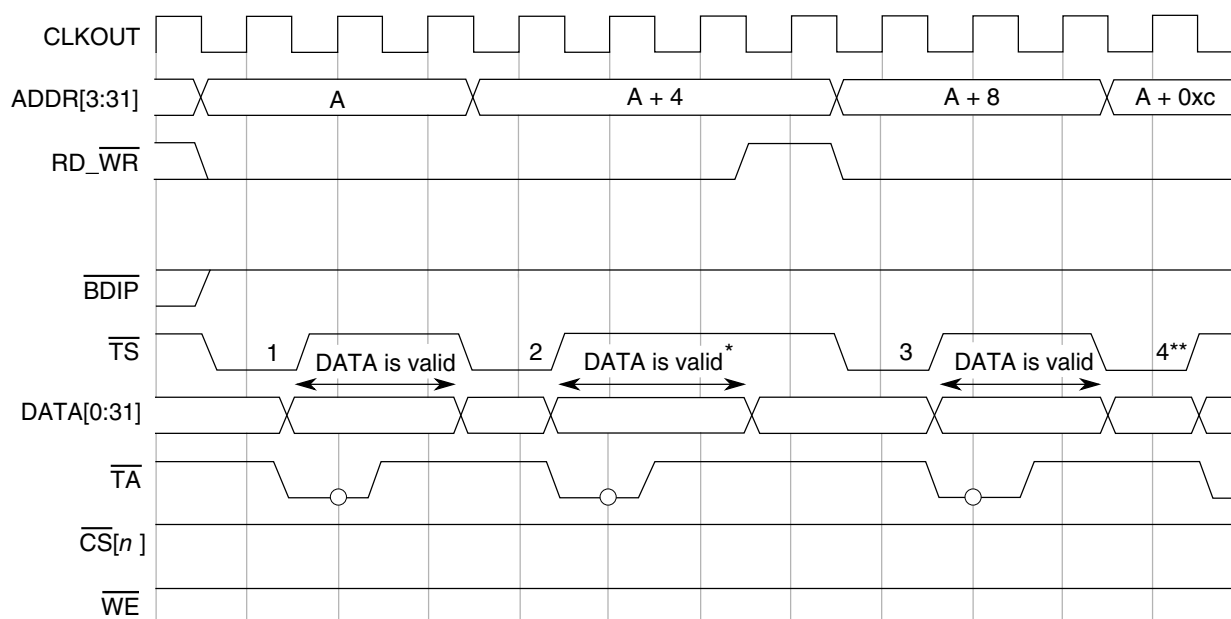


Figure 27-42. Single Beat 32-bit Write Cycle, 16-bit Port Size, Basic Timing

27.5.2.6.2 Small Access Example #2: 32-byte Write with External $\overline{TA}$

The following figure shows an example of a 32-byte write to a non-chip-select device using external $\overline{TA}$, requiring eight 32-bit external transactions. Note that due to the use of external $\overline{TA}$, $\overline{RD\_WR}$ does not toggle between the accesses unless that access is the end of a 64-bit boundary. In this case, an extra cycle is required between $\overline{TA}$ and the next $\overline{TS}$ in order to get the next 64-bits of write data internally and $\overline{RD\_WR}$ negates during this extra cycle.



* This extra cycle is required after accesses 2, 4, and 6 to get the next 64-bits of internal write data.

**Four more external accesses (not shown) are required to complete the internal 32-byte request. The timing of these is the same as accesses 1-4 shown in this diagram.

Figure 27-43. 32-Byte Write Cycle with External $\overline{TA}$, Basic Timing

27.5.2.6.3 Small Access Example #3: 32-byte Read to 32-bit Port with BL=1

Figure 27-44 shows an example of a 32-byte read to a 32-bit burst enabled port with burst length of 4 words, requiring two 4-word (16-byte) external transactions. For this case, the address for the 2nd 4-word burst access is calculated by adding 0x10 to the lower 5 bits of the 1st address (no carry), and then masking out the lower 4 bits to fix them at zero.

Table 27-39. Examples of 4-word Burst Addresses

1st Address	Lower 5 bits of 1st Address + 0x10 (no carry)	Final 2nd Address (After Masking Lower 4 Bits)
0x000	0x10	0x10
0x008	0x18	0x10
0x010	0x00	0x00
0x018	0x08	0x00
0x020	0x30	0x30
0x028	0x38	0x30
0x030	0x20	0x20
0x038	0x28	0x20

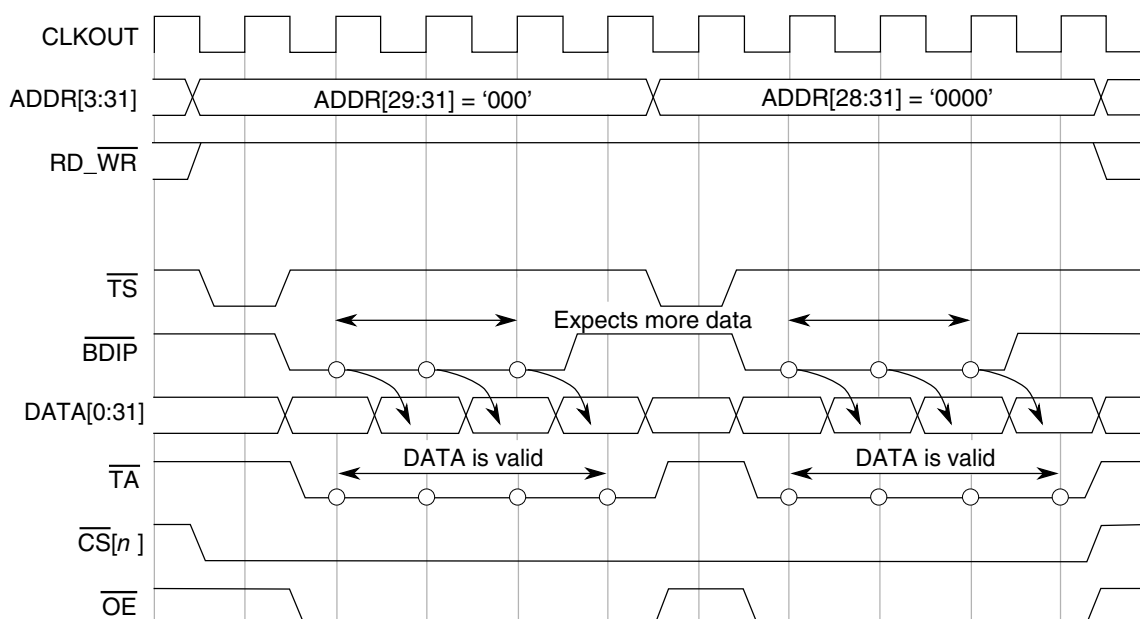
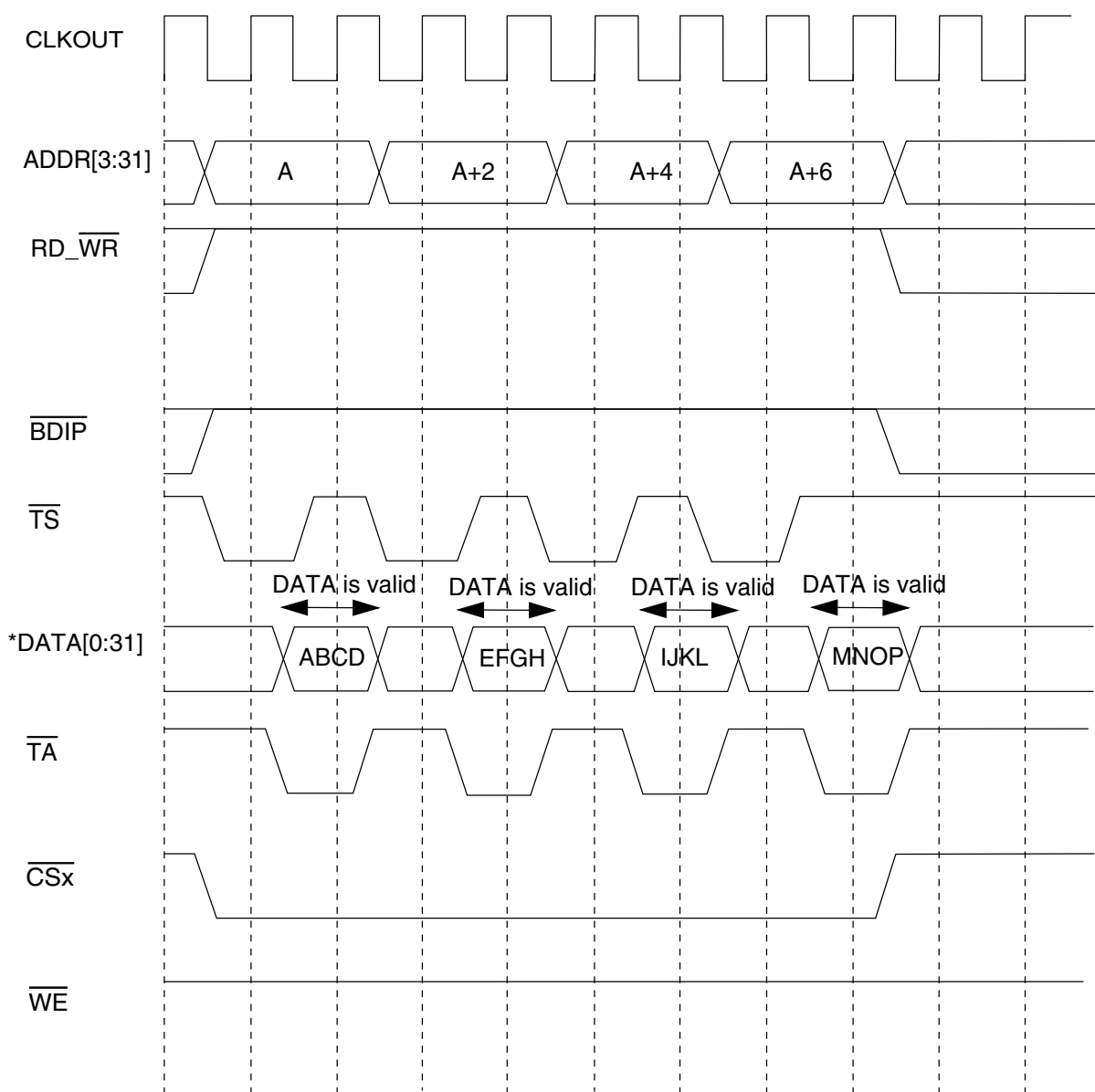


Figure 27-44. 32-Byte Read with B-T-B 4-Word Bursts to 32-bit Port, Zero Wait States

27.5.2.6.4 Small Access Example #4: 64-bit Read to 16-bit Port

The following figure shows an example of a 64-bit read to a 16-bit port, requiring four 16-bit external transactions.



* Or DATA[16:31], based on D16_31 bit in EBI_MCR.

Figure 27-45. Single Beat 64-bit Read Cycle, 16-bit Port Size, Basic Timing

27.5.2.6.5 Small Access Example #5: 32-byte Read to 32-bit Port with SBL=1

Figure 27-44 shows an example of a 32-byte read to a 32-bit burst enabled port with burst length of 2 words, requiring four 8-byte external transactions. For this case, the address for the 2nd-4th 2-word burst accesses is calculated by adding 0x08 to the lower 5 bits of the 1st address (no carry), and then masking out the lower 3 bits to fix them at zero.

Table 27-40. Examples of 2-word Burst Addresses

1st Address	Lower 5 bits of 1st Address + 0x08 (no carry)	Final 2nd-4th Addresses (After Masking Lower 3 Bits)
0x000	0x08	0x08, 0x10, 0x18
0x008	0x10	0x10, 0x18, 0x00
0x010	0x18	0x18, 0x00, 0x08
0x018	0x00	0x00, 0x8, 0x10
0x020	0x28	0x28, 0x30, 0x38
0x028	0x30	0x30, 0x38, 0x20
0x030	0x38	0x38, 0x20, 0x28
0x038	0x20	0x20, 0x28, 0x30

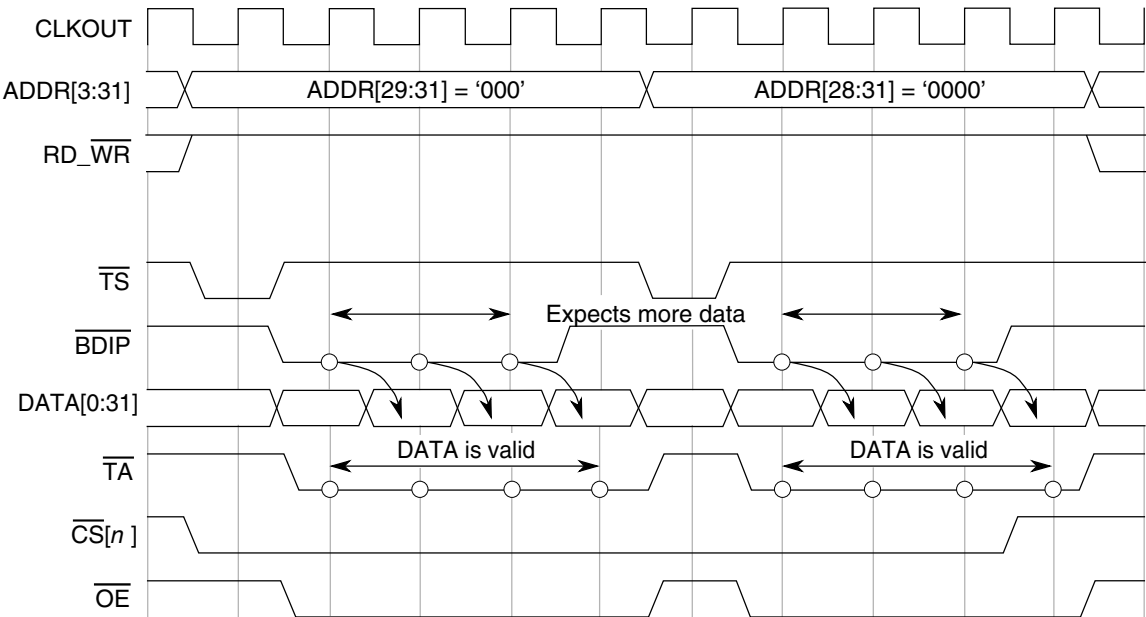


Figure 27-46. 32-Byte Read with 4 B-T-B 2-Word Bursts to 32-bit Port, Zero Wait States (SBL=1)

27.5.2.7 Size, Alignment and Packaging on Transfers

Table 27-41 shows the allowed sizes that an internal master can request from the EBI. The behavior of the EBI for request sizes not shown below is undefined. No error signal is asserted for these erroneous cases.

Table 27-41. Transaction Sizes Supported by EBI

# Bytes (internal master)	# Bytes (external master)
1	1
2	2

Table continues on the next page...

Table 27-41. Transaction Sizes Supported by EBI (continued)

# Bytes (internal master)	# Bytes (external master)
4	4
3 ¹	
8	
32 ²	

1. Some misaligned access cases may result in 3-byte writes. These cases are treated as power-of-2 sized requests by the EBI, using `WE_BE[0:3]` to make sure only the appropriate 3 bytes get written.
2. Only supported for case of 64-bit internal AMBA data bus.

Even though misaligned non-burst transfers from internal masters are supported, the EBI naturally aligns the accesses when it sends them out to the external bus, splitting them into multiple aligned accesses if necessary. See [Misaligned Access Support](#) for these cases.

Natural alignment for the EBI means:

- Byte access can have any address
- 16-bit access, address bit 31 must be 0
- 32-bit access, address bits 30-31 must be 0
- For burst accesses of any size, address bits 29-31 must be 0

The EBI never generates a misaligned external access, so a multi-master system with two e200-based MCUs can never have a misaligned external access from one to the other. In the erroneous case that an externally-initiated misaligned access does occur, the EBI errors the access (by asserting `TEA` externally) and does not initiate the access on the internal bus.

The EBI requires that the portion of the data bus used for a transfer to/from a particular port size be fixed. A 32-bit port must reside on data bus bits 0-31, and a 16-bit port must reside on bits 0-15.

In the following figures and tables the following convention is adopted:

- The most significant byte of a 32-bit operand is OP0, and OP3 is the least significant byte.
- The two bytes of a 16-bit operand are OP0 (most significant) and OP1, or OP2 (most significant) and OP3, depending on the address of the access.
- The single byte of a byte-length operand is OP0, OP1, OP2, or OP3, depending on the address of the access.

This can be seen in [Figure 27-47](#).

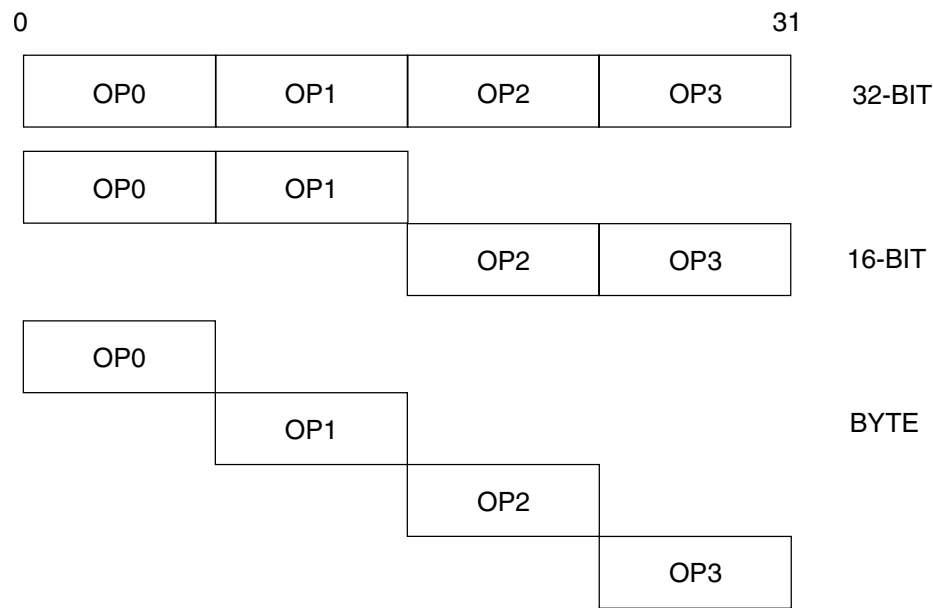


Figure 27-47. Internal Operand Representation

[Figure 27-48](#) shows the device connections on the DATA[0:31] bus.

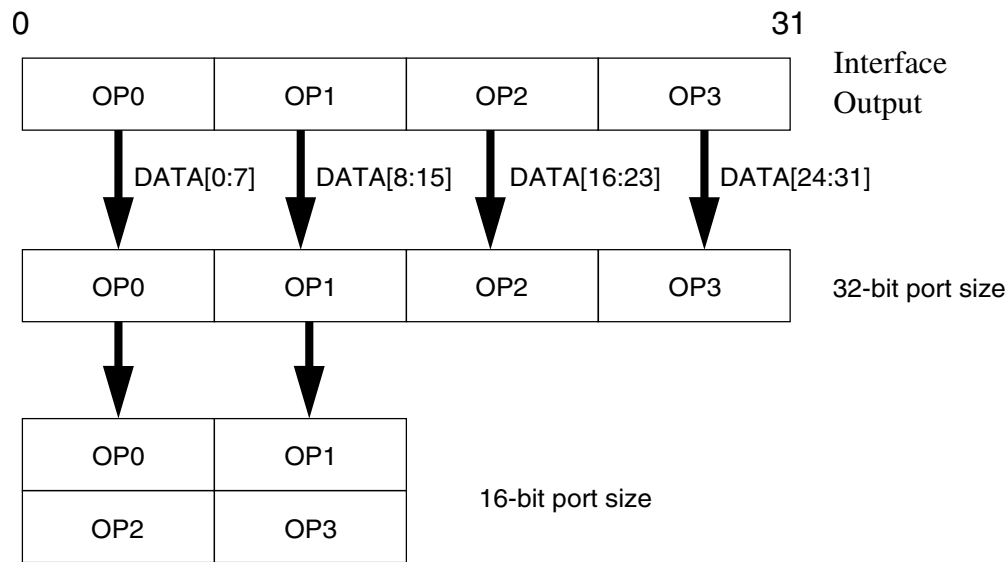


Figure 27-48. Interface to Different Port Size Devices

[Table 27-42](#) lists the bytes required on the data bus for read cycles. The bytes indicated as ‘-’ are not required during that read cycle.

[Table 27-43](#) lists the patterns of the data transfer for write cycles when accesses are initiated by the MCU. The bytes indicated as ‘-’ are not driven during that write cycle.

Table 27-42. Data Bus Requirements for Read Cycles

Transfer Size	TSIZ[0:1]	Address		32-Bit Port Size				16-Bit Port Size ¹	
		A30	A31	D0:D7	D8:D15	D16:D23	D24:D31	D0:D7 ²	D8:D15 ³
Byte	01	0	0	OP0	-	-	-	OP0	-
	01	0	1	-	OP1	-	-	-	OP1
	01	1	0	-	-	OP2	-	OP2	-
	01	1	1	-	-	-	OP3	-	OP3
16-bit	10	0	0	OP0	OP1	-	-	OP0	OP1
	10	1	0	-	-	OP2	OP3	OP2	OP3
32-bit	00	0	0	OP0	OP1	OP2	OP3	OP0/OP2 ⁴	OP1/OP3

1. Also applies when DBM=1 for 16-bit data bus mode.
2. For address/data muxed transfers, DATA[16:23] are used externally, not DATA[0:7].
3. For address/data muxed transfers, DATA[24:31] are used externally, not DATA[8:15].
4. This case consists of two 16-bit external transactions, the first fetching OP0 and OP1, the second fetching OP2 and OP3.

Table 27-43. Data Bus Contents for Write Cycles

Transfer Size	TSIZ[0:1]	Address		32-Bit Port Size				16-Bit Port Size ¹	
		A30	A31	D0:D7	D8:D15	D16:D23	D24:D31	D0:D7 ²	D8:D15 ³
Byte	01	0	0	OP0	-	-	-	OP0	-
	01	0	1	-	OP1	-	-	-	OP1
	01	1	0	-	-	OP2	-	OP2	-
	01	1	1	-	-	-	OP3	-	OP3
16-bit	10	0	0	OP0	OP1	-	-	OP0	OP1
	10	1	0	-	-	OP2	OP3	OP2	OP3
32-bit	00	0	0	OP0	OP1	OP2	OP3	OP0/OP2 ⁴	OP1/OP3

1. Also applies when DBM=1 for 16-bit data bus mode.
2. For address/data muxed transfers, DATA[16:23] are used externally, not DATA[0:7].
3. For address/data muxed transfers, DATA[24:31] are used externally, not DATA[8:15].
4. This case consists of two 16-bit external transactions, the first writing OP0 and OP1, the second writing OP2 and OP3.

27.5.2.8 Termination Signals Protocol

The termination signals protocol was defined in order to avoid electrical contention on lines that can be driven by various sources. In order to do that, a slave must not drive signals associated with the data transfer until the address phase is completed and it recognizes the address as its own. The slave must disconnect from signals immediately after it acknowledges the cycle and not later than the termination of the next address phase cycle.

For EBI-mastered non-chip-select accesses, the EBI requires assertion of $\overline{TA}$ from an external device to signal that the bus cycle is complete. The EBI uses a latched version of $\overline{TA}$ (1 cycle delayed) for these accesses to help make timing at high frequencies. This

results in the EBI driving the address and control signals 1 cycle longer than required, as seen in [Figure 27-49](#). However, the DATA does not need to be held 1 cycle longer by the slave, because the EBI latches DATA every cycle during non-chip-select accesses. During these accesses, the EBI does not drive the $\overline{TA}$ signal, leaving it up to an external device (or weak internal pullup) to drive $\overline{TA}$.

For EBI-mastered chip-select accesses, when the SETA bit is 0, the EBI drives $\overline{TA}$ the entire cycle, asserting according to internal wait state counters to terminate the cycle. When the SETA bit is 1, the EBI samples the $\overline{TA}$ for the entire cycle. During idle periods on the external bus, the EBI drives $\overline{TA}$ negated as long as it is granted the bus; when it no longer owns the bus, it lets go of $\overline{TA}$.

If no device responds by asserting $\overline{TA}$ within the programmed timeout period (BMT in EBI_BMCR) after the EBI initiates the bus cycle, the internal Bus Monitor (if enabled) asserts $\overline{TEA}$ to terminate the cycle. An external device may also drive $\overline{TEA}$ when it detects an error on an external transaction. $\overline{TEA}$ assertion causes the cycle to terminate and the processor to enter exception processing for the error condition. To properly control termination of a bus cycle for a bus error with external circuitry, $\overline{TEA}$ must be asserted at the same time or before (external) $\overline{TA}$ is asserted. $\overline{TEA}$ must be negated before the second rising edge after it was sampled asserted in order to avoid the detection of an error for the following bus cycle initiated. $\overline{TEA}$ is only driven by the EBI during the cycle where the EBI is asserting $\overline{TEA}$ and the cycle immediately following this assertion (for fast negation). During all other cycles, the EBI relies on a weak internal pullup to hold $\overline{TEA}$ negated. This allows an external device to assert $\overline{TEA}$ when it needs to indicate an error. External devices must follow the same protocol as the EBI, only driving $\overline{TEA}$ during the assertion cycle and 1 cycle afterwards for negation.

When $\overline{TEA}$ is asserted from an external source, the EBI uses a latched version of $\overline{TEA}$ (1 cycle delayed) to help make timing at high frequencies. This means that for any accesses where the EBI drives $\overline{TA}$ (chip-select accesses with SETA=0), a $\overline{TEA}$ assertion that occurs 1 cycle before or during the last $\overline{TA}$ of the access could be ignored by the EBI, since it will have completed the access internally before it detects the latched $\overline{TEA}$ assertion. This means that non-burst chip-select accesses with no wait states (SCY=0) cannot be reliably terminated by external $\overline{TEA}$. If external error termination is required for such a device, the EBI must be configured for SCY>=1.

Note

For the cases discussed above where $\overline{TEA}$ "could be ignored", this is not guaranteed. For some small access cases (which always use chip-select and internally-driven $\overline{TA}$), a $\overline{TEA}$ that occurs 1 cycle before or during the $\overline{TA}$ cycle or for SCY=0 may in fact lead to terminating the cycle with error. However, proper error termination is not guaranteed for these cases, so $\overline{TEA}$

must always be asserted at least 2 cycles before an internally-driven $\overline{TA}$ cycle for proper error termination.

External $\overline{TEA}$ assertion that occurs during the same cycle that $\overline{TS}$ is asserted by the EBI is always treated as an error (terminating the access) regardless of SCY.

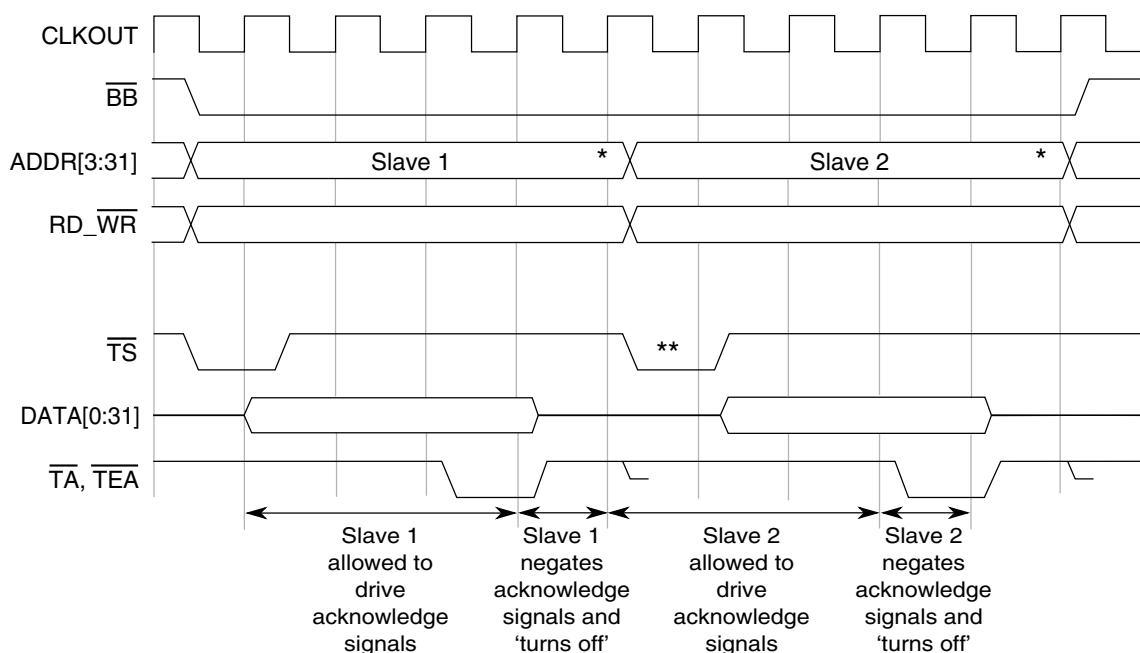
The following table summarizes how the EBI recognizes the termination signals provided from an external device.

Table 27-44. Termination Signals Protocol

$\overline{TEA}^{1,1}$	$\overline{TA}^1$	Action
Negated	Negated	No Termination
Asserted	X	Transfer Error Termination
Negated	Asserted	Normal Transfer Termination

1. Latched version (1 cycle delayed) used for externally driven $\overline{TEA}$ and $\overline{TA}$.

The following figure shows an example of the termination signals protocol for back-to-back reads to two different slave devices who properly "take turns" driving the termination signals. This assumes a system using slave devices that drive termination signals.



* The EBI drives address and control signals an extra cycle because it uses a latched version of $\overline{TA}$ (1 cycle delayed) to terminate the cycle. An external master is not required to do this.

**This is the earliest that the EBI can start another transfer, in the case of continuing a set of small accesses. For all other cases, an extra cycle is needed before the EBI can start another $\overline{TS}$.

Figure 27-49. Termination Signals Protocol Timing Diagram

27.5.2.9 Non-Chip-Select Burst in 16-bit Data Bus Mode

The timing diagrams in this section apply only to the special case of a non-chip-select 32-bit access in 16-bit data bus mode (DBM=1 in EBI_MCR).

For this case, a special 2-beat burst protocol is used for reads and writes, so that a slave device (using the same EBI) can internally generate one 32-bit read or write access (thus 32-bit coherent), as opposed to two separate 16-bit accesses.

Note

If the device does not support multi-master systems, the original intent of this protocol does not apply. However, this 2-beat burst protocol can also occur in a single-master system, if a non-chip-select 32-bit access to a 16-bit port is performed.

Figure 27-50 shows a 32-bit (and non-chip-select in a single-master system) read from an external master in 16-bit data bus mode.

Figure 27-51 shows a 32-bit (and non-chip-select in a single-master system) write from an external master in 16-bit data bus mode.

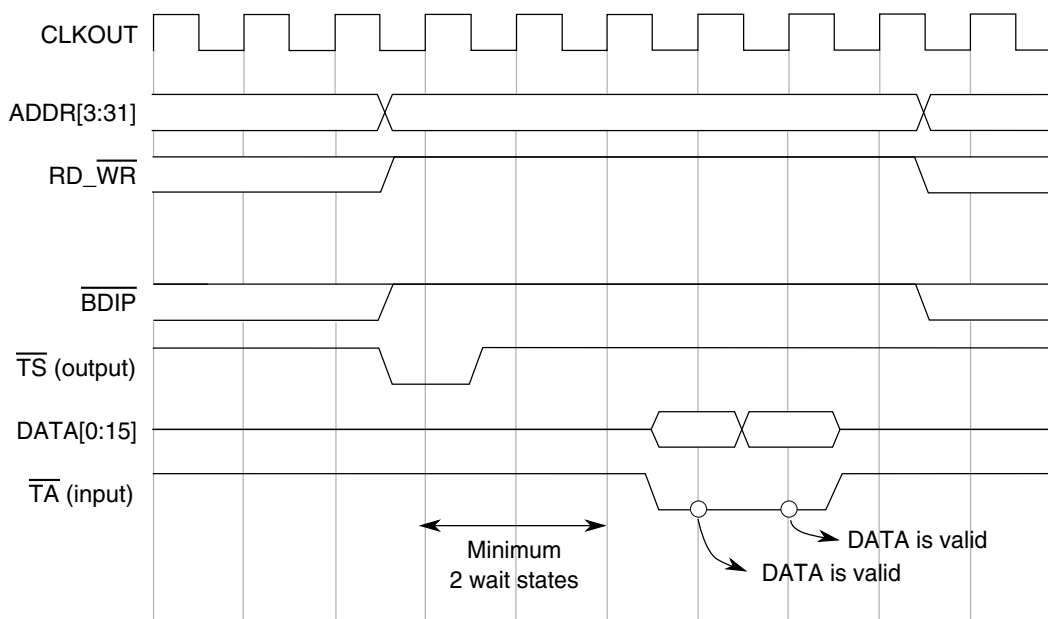


Figure 27-50. 32-bit Read from MCU with DBM=1

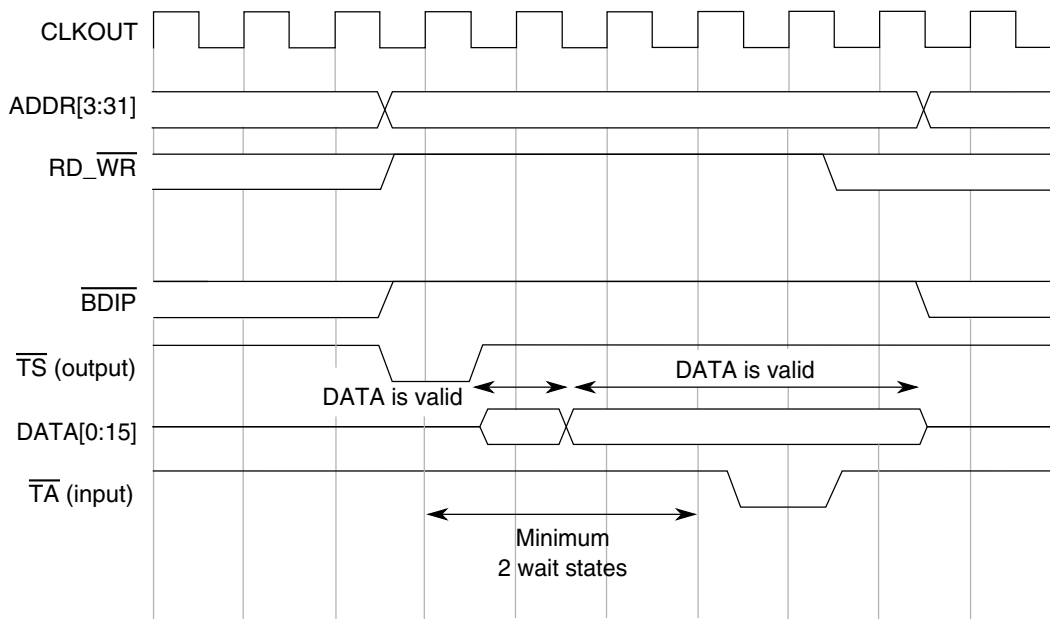


Figure 27-51. 32-bit Write to MCU with DBM=1

27.5.2.10 Calibration Bus Operation

Some devices with this EBI have a second external bus, intended for calibration use. This bus consists of a second set of the same signals present on the Primary external bus, except that arbitration, (and optionally other signals also) are excluded. Both busses can be supported with one EBI block, by using the calibration chip-selects to steer accesses to the calibration bus instead of the primary external bus.

Since the calibration bus has no arbitration signals, the arbitration on the primary bus controls accesses on the calibration bus as well, and no external master accesses can be performed on the calibration bus. Accesses cannot be performed in parallel on both external busses. However, back-to-back accesses can switch from one bus to the other, as determined by the type of chip-select each address matches.

The timing diagrams and protocol for the calibration bus is identical to the primary bus, except that some signals are missing on the calibration bus. See the device-specific documentation for the calibration bus signal list for a particular MCU.

There is an inherent dead cycle between a calibration chip-select access and a non-calibration access (chip-select or non-chip-select), just like the one between accesses to two different non-calibration chip-selects (described in [Back-to-Back Accesses](#)).

Figure 27-52 shows an example of a non-calibration chip-select read access followed by a calibration chip-select read access. Note that this figure is identical to Figure 27-28, except the CSy is replaced by CAL_CSy. Timing for other cases on calibration bus can similarly be derived from other figures in this document (by replacing $\overline{\text{CS}}$ with $\overline{\text{CAL\_CS}}$).

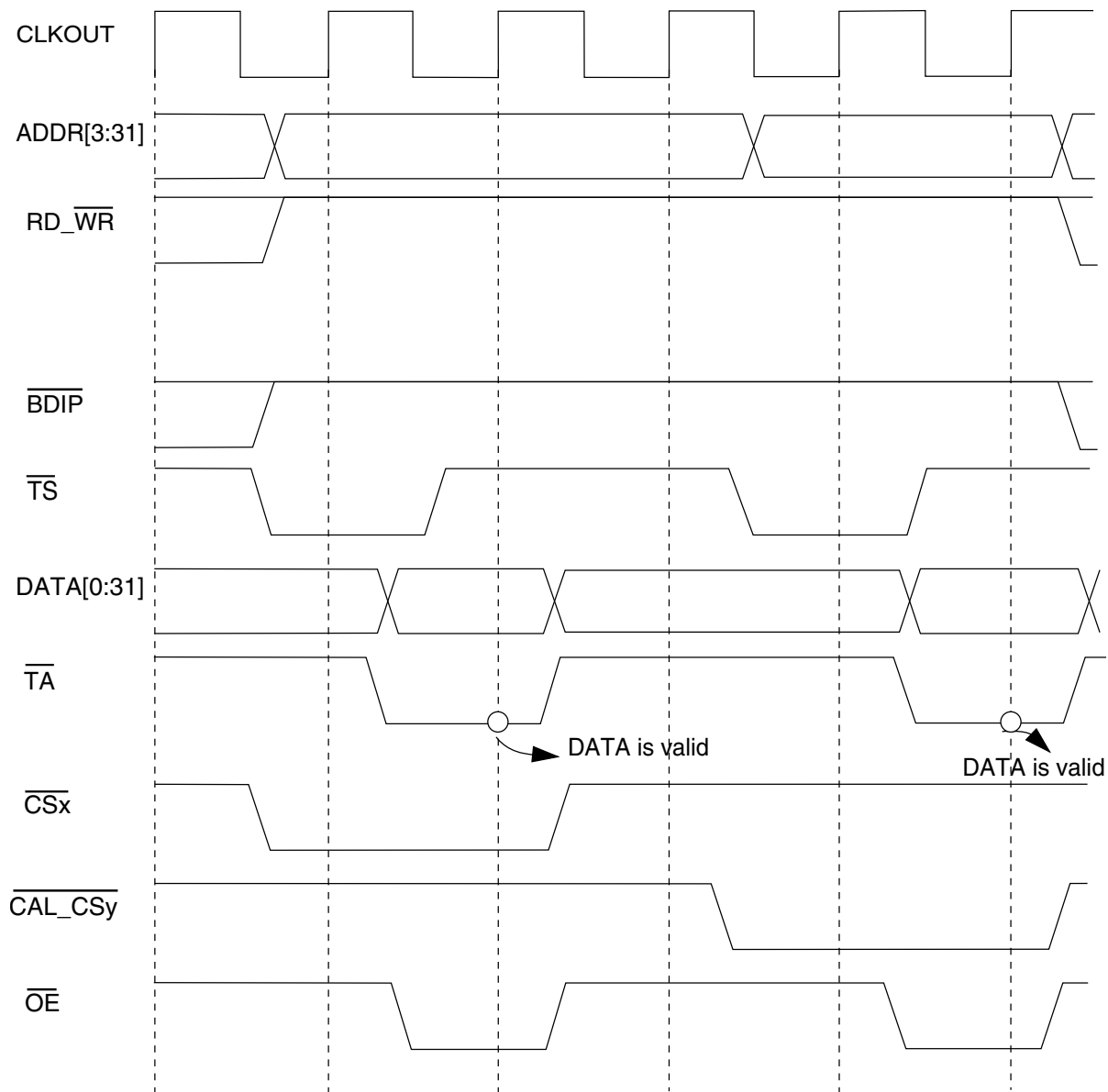


Figure 27-52. Back-to-Back 32-bit Reads to $\overline{\text{CS}}$, $\overline{\text{CAL\_CS}}$ Banks

27.5.2.11 Misaligned Access Support

This section describes all the misaligned cases supported by the EBI. These cases are a subset of the full set of cases allowed by the AMBA AHB V6 specification. The EBI works under the assumption that all internal masters on the device do not produce any misaligned access cases (to the EBI) other than the ones below.

27.5.2.11.1 Misaligned Access Support (64 bit AMBA)

Table 27-45 shows all the misaligned access cases supported by the EBI (using a 64-bit AMBA implementation), as seen on the internal master AMBA bus. All other misaligned cases are not supported. If an unsupported misaligned access to the EBI is attempted (such as non-chip-select or burst misaligned access), the EBI errors the access on the internal bus and does not start the access (nor assert $\overline{TEA}$) externally.

Table 27-45. Misalignment Cases Supported by a 64 bit AMBA EBI (internal bus)

No. ¹	Program Size and byte offset	Address [29:31] ²	Data Bus Byte Strobes ³	HSIZE ⁴	HUNALIGN ⁵
1	Half @0x1	001	0110_0000	10	1
2	Half @0x3	011	0001_1000	11 ^{6, 6}	1
3	Half @0x3	011	0001_0000	01 ^{7, 7}	1
-	(2 AHB transfers)	100	0000_1000	00	0
4	Half @0x5	101	0000_0110	10	1
5	Half @0x7	111	0000_0001	01 ⁷	1
-	(2 AHB transfers)	000	1000_0000	00	0
6	Word @0x1	001	0111_1000	11	1
7	Word @0x1	001	0111_0000	10	1
-	(2 AHB transfers)	100	0000_1000	00	0
8	Word @0x2	010	0011_1100	11 ⁶	1
9	Word @0x2	010	0011_0000	10 ⁸	1
-	(2 AHB transfers)	100	0000_1100	01	0
10	Word @0x3	011	0001_1110	11	1
11	Word @0x3	011	0001_0000	10 ⁷	1
12	(2 AHB transfers)	100	0000_1110	10	1
13	Word @0x5	101	0000_0111	10	1
-	(2 AHB transfers)	000	1000_0000	00	0
14	Word @0x6	110	0000_0011	10 ^{8, 8}	1
-	(2 AHB transfers)	000	1100_0000	01	0
15	Word @0x7	111	0000_0001	10 ⁷	1
16	(2 AHB transfers)	000	1110_0000	10	1

Table continues on the next page...

**Table 27-45. Misalignment Cases Supported by a 64 bit AMBA EBI (internal bus)
(continued)**

No. ¹	Program Size and byte offset	Address [29:31] ²	Data Bus Byte Strobes ³	HSIZE ⁴	HUNALIGN ⁵
17	Doubleword @0x1	001	0111_1111	11	1
-	(2 AHB transfers)	000	1000_0000	00	0
18	Doubleword @0x2	010	0011_1111	11	1
-	(2 AHB transfers)	000	1100_0000	01	0
19	Doubleword @0x3	011	0001_1111	11	1
20	(2 AHB transfers)	000	1110_0000	10	1
21	Doubleword 0x4	100	0000_1111	11 ^{9, 9}	1
-	(2 AHB transfers)	000	1111_0000	10	0
22	Doubleword @0x5	101	0000_0111	11 ⁹	1
23	(2 AHB transfers)	000	1111_1000	11	1
24	Doubleword @0x6	110	0000_0011	11 ⁸	1
25	(2 AHB transfers)	000	1111_1100	11	1
26	Doubleword 0x7	111	0000_0001	11 ⁷	1
27	(2 AHB transfers)	000	1111_1110	11	1

1. Misaligned case number. Only transfers where HUNALIGN=1 are numbered as misaligned cases.
2. Address on internal master AHB bus, not necessarily address on external ADDR pins.
3. Internal byte strobe signals on AHB bus. Shown with Big-Endian byte ordering in this table, even though internal master AHB bus uses Little-Endian byte-ordering (EBI flips order internally).
4. Internal signal on AHB bus; 00=8-bits, 01=16 bits, 10=32 bits, 11=64-bits. HSIZE is driven according to the smallest aligned container that contains all the requested bytes. This results in extra EBI external transfers in some cases.
5. Internal signal on AHB bus that indicates that this transfer is misaligned (when 1).
6. For this case, the EBI internally treats HSIZE as 10 (4-byte access), if PS=1 (16-bit port).
7. For this case, the EBI internally treats HSIZE as 00(1-byte access).
8. For this case, the EBI internally treats HSIZE as 01 (2-byte access).
9. For this case, the EBI internally treats HSIZE as 10 (4-byte access).

Table 27-46 shows which external transfers are generated by the EBI for the misaligned access cases in **Table 27-45**, for each port size.

The number of external transfers for each internal AHB master request is determined by the HSIZE value for that request (adjusted internally in some cases, see table footnotes) relative to the port size. For example, a Word write to @011 with 16-bit port size (case #10 in **Table 27-45**) results in 4 external 16-bit transfers because the HSIZE is 64-bits. For cases where two or more external transfers are required for one internal transfer request, these external accesses are considered part of a "small access" set, as described in **Small Accesses (Small Port Size and Short Burst Length)**.

Since all transfers are aligned on the external bus, normal timing diagrams and protocol apply. Note that the TSIZ[0:1] signals are not intended to be used for misaligned accesses, so they are not specified in **Table 27-46**.

Table 27-46. Misalignment Cases Supported by a 64 bit AMBA EBI (external bus)

No. ¹	PS ²	Program Size and byte offset	ADDR[29:31] ³	WE_BE[0:3] ⁴
1	0	Half @0x1	000	1001
	1		000	1011
			010	0111
2	0	Half @0x3	000	1110
	1		100	0111
			010	1011
3	0	Half @0x3 (2 AHB transfers)	100	0111
			011	1011
	1		100	0111
			-	-
4	0	Half @0x5	100	1001
	1		100	1011
			110	0111
5	0	Half @0x7(2 AHB transfers)	111	1110
-	000		0111	
5	1		111	1011
-			000	0111
6	0	Word @0x1	000	1000
			100	0111
	1		000	1011
			010	0011
			100	0111
7	0	Word @0x1(2 AHB transfers)	000	1000
-	100		0111	
7	1		000	1011
			010	0011
-	100		0111	
8	0	Word @0x2	000	1100
	1		100	0011
			010	0011
			100	0011
9	0	Word @0x2(2 AHB transfers)	000	1100
-	100		0011	
9	1		010	0011
			100	0011

Table continues on the next page...

**Table 27-46. Misalignment Cases Supported by a 64 bit AMBA EBI (external bus)
(continued)**

No. ¹	PS ²	Program Size and byte offset	ADDR[29:31] ³	WE_BE[0:3] ⁴
10	0	Word @0x3	000	1110
			100	0001
	1		010	1011
			100	0011
			110	0111
11	0	Word @0x3(2 AHB transfers)	000	1110
12			100	0001
11	1		010	1011
			100	0011
			110	0111
13	0	Word @0x5(2 AHB transfers)	100	1000
-			000	0111
13	1		100	1011
			110	0011
-	000		0111	
14	0	Word @0x6(2 AHB transfers)	110	1100
-			000	0011
14	1		110	0011
			000	0011
15	0		Word @0x7(2 AHB transfers)	111
16		000		0001
15	1	111		1011
		000		0011
16	010	0111		
17	0	Doubleword @0x1 (2 AHB transfers)	000	1000
			100	0000
	-		000	0111
17	1		000	1011
			010	0011
		100	0011	
		110	0011	
-			000	0111

Table continues on the next page...

**Table 27-46. Misalignment Cases Supported by a 64 bit AMBA EBI (external bus)
(continued)**

No. ¹	PS ²	Program Size and byte offset	ADDR[29:31] ³	WE_BE[0:3] ⁴
18	0	Doubleword @0x2 (2 AHB transfers)	000	1100
-			100	0000
18	1		000	0011
-			010	0011
-			100	0011
-			110	0011
19	0	Doubleword @0x3 (2 AHB transfers)	000	1110
20			100	0000
19	1		000	0001
-			010	1011
-			100	0011
-			110	0011
20	-	000	0011	
-	-	010	0111	
21	0	Doubleword @0x4 (2 AHB transfers)	100	0000
-			000	0000
21	1		100	0011
-			110	0011
-			000	0011
-			010	0011
22	0	Doubleword @0x5 (2 AHB transfers)	100	1000
23			000	0000
22	1		100	0111
-			100	1011
-			110	0011
-			000	0011
23	-	010	0011	
-	-	100	0111	
24	0	Doubleword @0x6 (2 AHB transfers)	110	1100
25			000	0000
24	1		100	0011
-			110	0011
-			000	0011
-			010	0011
25	-	100	0011	

Table continues on the next page...

**Table 27-46. Misalignment Cases Supported by a 64 bit AMBA EBI (external bus)
(continued)**

No. ¹	PS ²	Program Size and byte offset	ADDR[29:31] ³	WE_BE[0:3] ⁴
26	0	Doubleword @0x7 (2 AHB transfers)	111	1110
27			000	0000
			100	0001
26	1		111	1011
27			000	0011
			010	0011
			100	0011
			110	0111

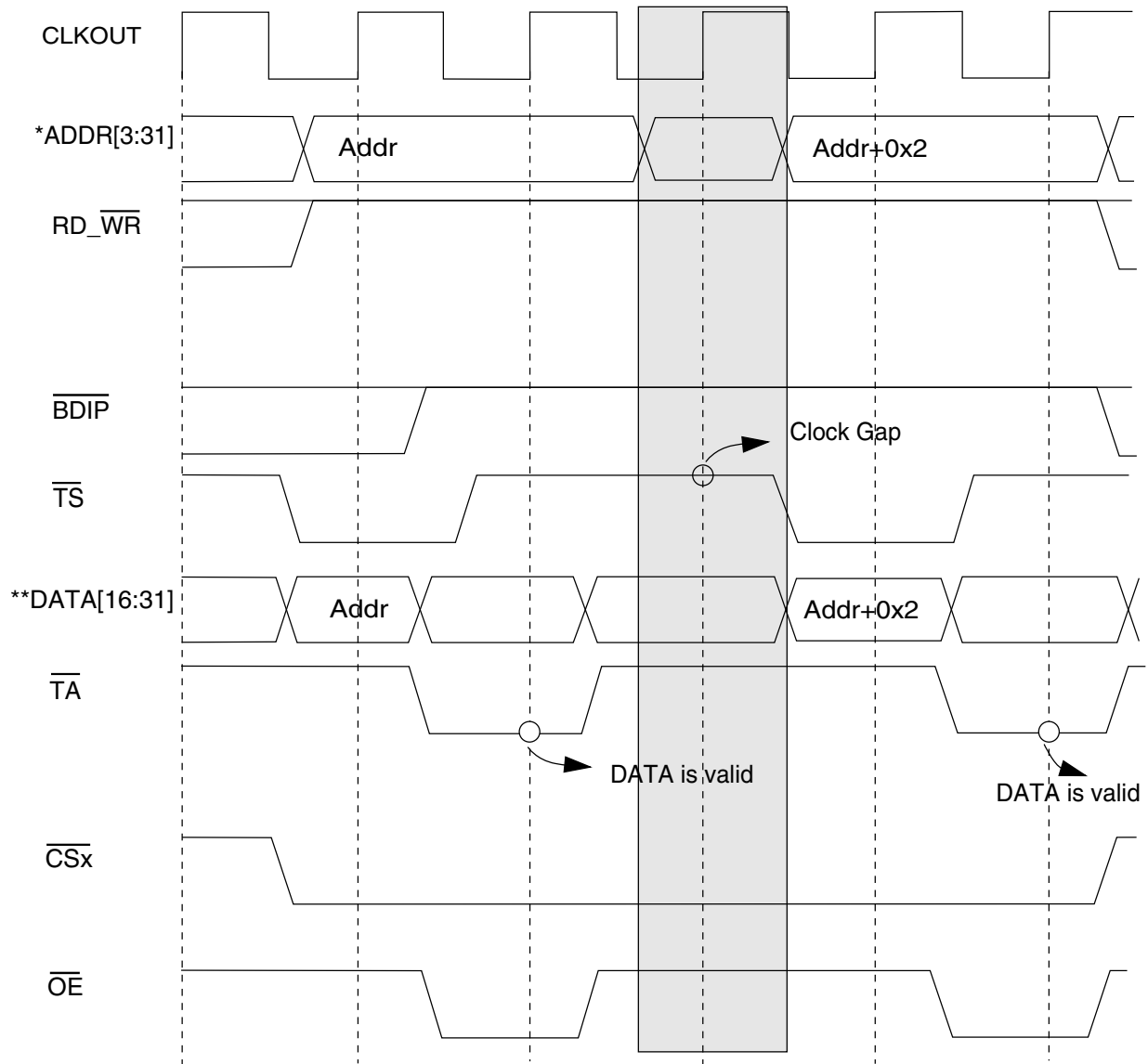
1. Misaligned case number, from [Table 27-45](#).
2. Port size; 0=32 bits, 1=16 bits.
3. External ADDR pins, not necessarily the address on internal master AHB bus.
4. External WE_BE pins. Note that these pins have negative polarity, opposite of the internal byte strobes in [Table 27-45](#).

27.5.2.12 Address Data Multiplexing

Address/Data multiplexing enables the design of a system with reduced pin count. In such a system, multiplexed address/data functions (on DATA pins) are used, instead of having separate address and data pins. Compared to the normal EBI specification (e.g. 24 address pins+32 data pins), only 32 data pins are required. Compared to a 16-bit bus implementation, only 24 pins are required (e.g. ADDR[8:15] + ADDR[16:31]/DATA[16:31]).

When performing a small access read, as described in [Small Accesses \(Small Port Size and Short Burst Length\)](#), with A/D multiplexing enabled for this access, the EBI inserts an idle clock cycle with $\overline{OE}$ negated and $\overline{CS}$ asserted (for GCSN=0), to allow for the memory to three-state the bus prior to the EBI driving the address on the next clock. This clock gap already exists (for other reasons) for non-small-access transfers, so no additional clock gap is inserted for those cases. See [Figure 27-53](#) for an example of a small access read with A/D multiplexing enabled.

In general, timing diagrams in A/D multiplexing mode are very similar to other diagrams in this document (including support for Burst accesses), except for the behavior of the ADDR and DATA busses, which can be seen in the following figure.



* While the EBI drives all of ADDR[3:31] to valid address, typically only ADDR[3:15] (or less) are used in the system, as DATA[16:31] (or DATA[0:15]) would be used for address and data on an external muxed device.

** Or DATA[0:15], based on D16_31 bit in EBI_MCR.

Figure 27-53. Small access (32-bit read to 16-bit port) on Address/Data multiplexed bus

27.5.2.13 Address / Write Enable Multiplexing

Address/Write Enable multiplexing enables the design of a system with reduced pin count (saves 4 pins). The typical usage is the case where the SoC is connected to an external chip-select device that requires write-enable functionality, but does not require the full address bus width. In such a system, the user can set EBI_OR[AWE]=1 to cause the EBI to drive its internal WE[0:3] signals onto the upper 4 ADDR pins, instead of relying on separate WE[0:3] pins (which may not be present on some SoCs).

In general, timing diagrams using ADDR/WE multiplexing (EBI_OR[AWE]=1) are very similar to other diagrams in this document, except for the behavior of the upper 4 ADDR signals, which can be seen in the following figure.

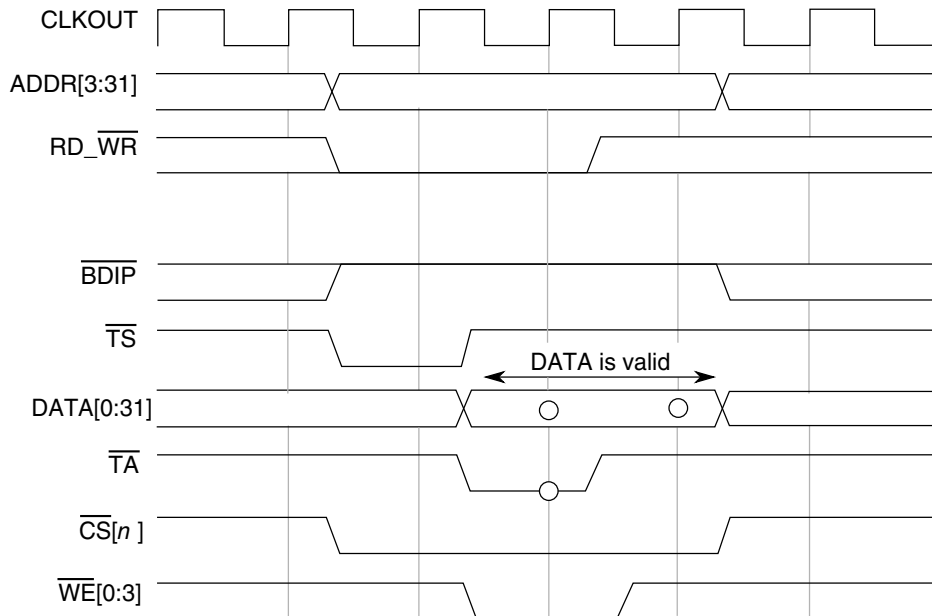


Figure 27-54. Single Beat 32-bit Write Cycle, $\overline{CS}$ Access, Zero Wait States, AWE=1

27.5.2.14 Address by Port Size

Address by Port Size (EBI_OR[APS]=1) provides a way to increase the memory size supported by the EBI for external chip-select devices with 16 or 32-bit port size.

For example, consider an SoC that has ADDR[8:31] pinned out and connects to an external 32-bit chip-select memory:

- If APS=0, then ADDR[31] is not connected to the memory and thus only 22 address lines (ADDR[8:29]) are left to connect to the memory.
- If APS=1, then the EBI drives internal address 6:29 to external ADDR[8:31], thus giving the user a 24-bit address range to access the 16-bit memory.

In general, timing diagrams using APS=1 are very similar to other diagrams in this document, except for the shifting of the ADDR signals, which can be seen in [Figure 27-55](#) and [Figure 27-56](#).

On some SoCs that use this EBI, the internal AMBA master is capable of requesting a WRAP4 32-bit burst even with a 64-bit AMBA bus width. This case is supported by the AMBA standard, but is rarely used on SoCs, since normally the full AMBA width is used when bursting (for optimum performance). A scenario where this case occurs is when there is a 32-bit AMBA master connecting through a 32->64 bit gasket that drives requests to the EBI's 64-bit AMBA bus.

In the 32-bit burst on 64-bit AMBA case, the EBI returns data on the appropriate half of the 64-bit AMBA bus, based on the address of the transfer (address bit 2 using N:0 numbering), and then alternates halves on each AMBA burst beat. Examples below are based on AMBA data bus numbering 63:0.

- For a burst starting at address 0x0 (or 0x8, 0x10, etc.), the 4 beats of data are returned on hrddata [63:32], [31:0], [63:32], [31:0].
- For a burst starting at address 0x4 (or 0xC, 0x14, etc.), the 4 beats of data are returned on hrddata [31:0], [63:32], [31:0], [63:32]

Note

For the case of a 32-bit burst on 64-bit AMBA where EBI_BR[BL]=0 and EBI_BR[SBL]=0, the external burst size (8 words) is mismatched to the internal burst size (4 words). In this case, to avoid erroneous data due to different wrapping boundaries, the EBI disables external burst for that access and splits the internal burst request to multiple single external accesses.

27.5.2.16 Internal AMBA Master Burst Read Abort Case

On some SoCs that use this EBI, the internal AMBA master has the capability to abort the AMBA burst read early, without performing the 4 requests that are normally part of an AMBA WRAP4 burst sequence. For example, instead of the usual htrans=NSEQ, SEQ, SEQ, SEQ, the master might do 1 of these cases:

- htrans=NSEQ (with hburst=WRAP4), SEQ (0-2 times), NSEQ (2nd NSEQ aborts prior burst)
- htrans=NSEQ (with hburst=WRAP4), SEQ (0-2 times), IDLE (IDLE aborts prior burst)

In these cases, the EBI has already started the burst externally, but will abort it early (without asserting $\overline{\text{TEA}}$ or internal hresp), and only returns the beats of data on the internal AMBA bus corresponding to the AMBA NSEQ/SEQ requests received. In a

subset of these master burst abort cases, there may be extra $\overline{\text{TS}}/\overline{\text{TA}}$ transfers on the external bus after the internal AMBA master has aborted (e.g. finishing a “small access” set of transfers). These extra read accesses are extraneous and can just be ignored.

This feature is not part of the AMBA standard, but is implemented in some AMBA masters for performance reasons. The master burst abort feature applies to reads only. The behavior of the EBI for an aborted AMBA burst write is undefined (not supported).

27.6 Initialization/Application Information

27.6.1 Booting from External Memory

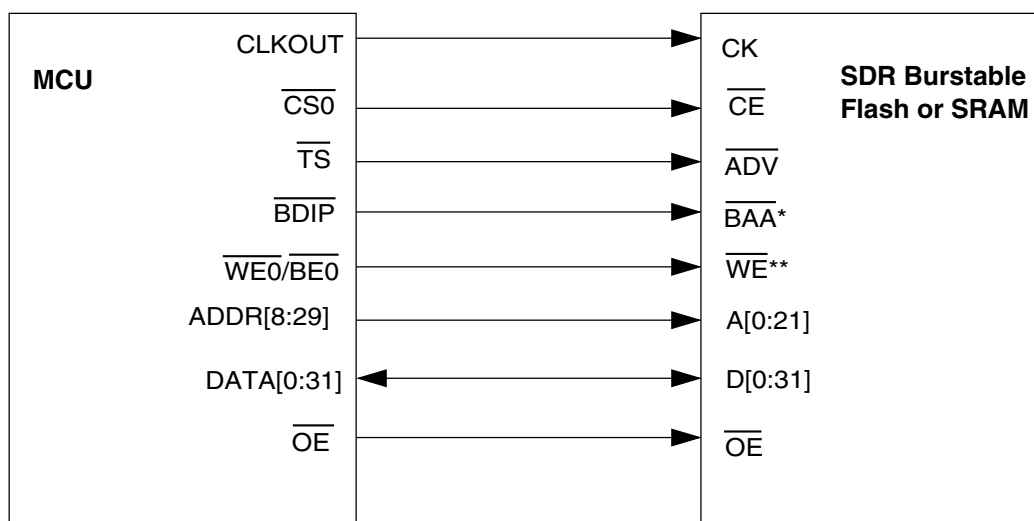
The EBI block does not support booting directly to external memory (i.e. fetching the first instruction after reset externally). One common method for an MCU to resemble an external boot with this EBI is to use an internal Boot Assist Module on the MCU, which fetches the first instruction internally and configures EBI registers before branching to an external address to “boot” externally. Refer to the device-specific documentation to see how/if external boot is supported for a particular MCU.

If code in external memory needs to write EBI registers, this must be done in a way that avoids modifying EBI registers while external accesses are being performed, such as the following method:

- Copy the code that is doing the register writes (plus a return branch) to internal SRAM
- Branch to internal SRAM to run this code, ending with a branch back to external flash

27.6.2 Running with SDR (Single Data Rate) Burst Memories

This includes FLASH and SRAM memories with a compatible burst interface. $\overline{\text{BDIP}}$ is required only for some SDR memories. The following figure shows a block diagram of an MCU connected to a 32-bit SDR burst memory.



* May or may not be connected, depending on the memory used.

** Flash memories typically use one $\overline{WE}$ signal as shown, RAMs use 2 or 4 (16-bit or 32-bit)

Figure 27-57. MCU Connected to SDR Burst Memory

Refer to [Figure 27-38](#) for an example of the timing of a typical Burst Read operation to an SDR burst memory. Refer to [Figure 27-22](#) for an example of the timing of a typical Single Write operation to SDR memory.

27.6.3 Running with Asynchronous Memories

The EBI also supports asynchronous memories. In this case, the CLKOUT, $\overline{TS}$, and $\overline{BDIP}$ pins are not used by the memory and bursting is not supported. However, the EBI still drives these outputs, and always drives and latches all signals at posedge CLKOUT (i.e., there is no "asynchronous mode" for the EBI). The data timing is controlled by setting the SCY bits in the appropriate Option Register to the proper number of wait states to work with the access time of the asynchronous memory, just as done for a synchronous memory.

27.6.3.1 Example Wait State Calculation

This example applies to any chip-select memory, synchronous or asynchronous.

As an example, say we have a memory with 50ns access time, and we are running the external bus @ 66 MHz (CLKOUT period: 15.2ns). Assume the input data spec for the MCU is 4ns.

number of wait states = (access time) / (CLKOUT period) + (0 or 1) (depending on setup time)

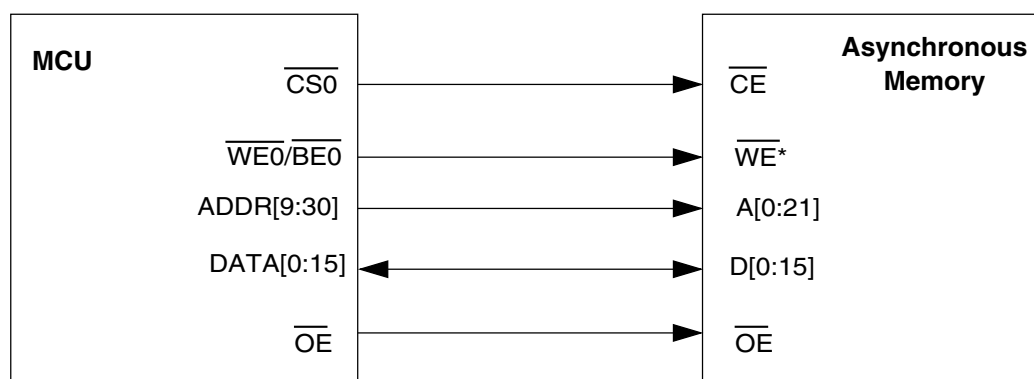
$50/15.2 = 3$ with 4.4ns remaining (so we need at least 3 wait states, now check setup time)

$15.2 - 4.4 = 10.8\text{ns}$ (this is the achieved input data setup time)

Since actual input setup (10.8ns) is greater than the input setup spec (4.0ns), 3 wait states is sufficient. If the actual input setup was less than 4.0ns, we would have to use 4 wait states instead.

27.6.3.2 Timing and Connections for Asynchronous Memories

The connections to an asynchronous memory are the same as for a synchronous memory, except that the CLKOUT, $\overline{\text{TS}}$, and $\overline{\text{BDIP}}$ signals are not used. Figure 27-58 shows a block diagram of an MCU connected to an asynchronous memory.



* Flash memories typically use one $\overline{\text{WE}}$ signal as shown, RAMs use 2 or 4 (16-bit or 32-bit)

Figure 27-58. MCU Connected to Asynchronous Memory

Figure 27-59 shows a timing diagram of a read operation to a 16-bit asynchronous memory using 3 wait states.

Figure 27-60 shows a timing diagram of a write operation to a 16-bit asynchronous memory using 3 wait states.

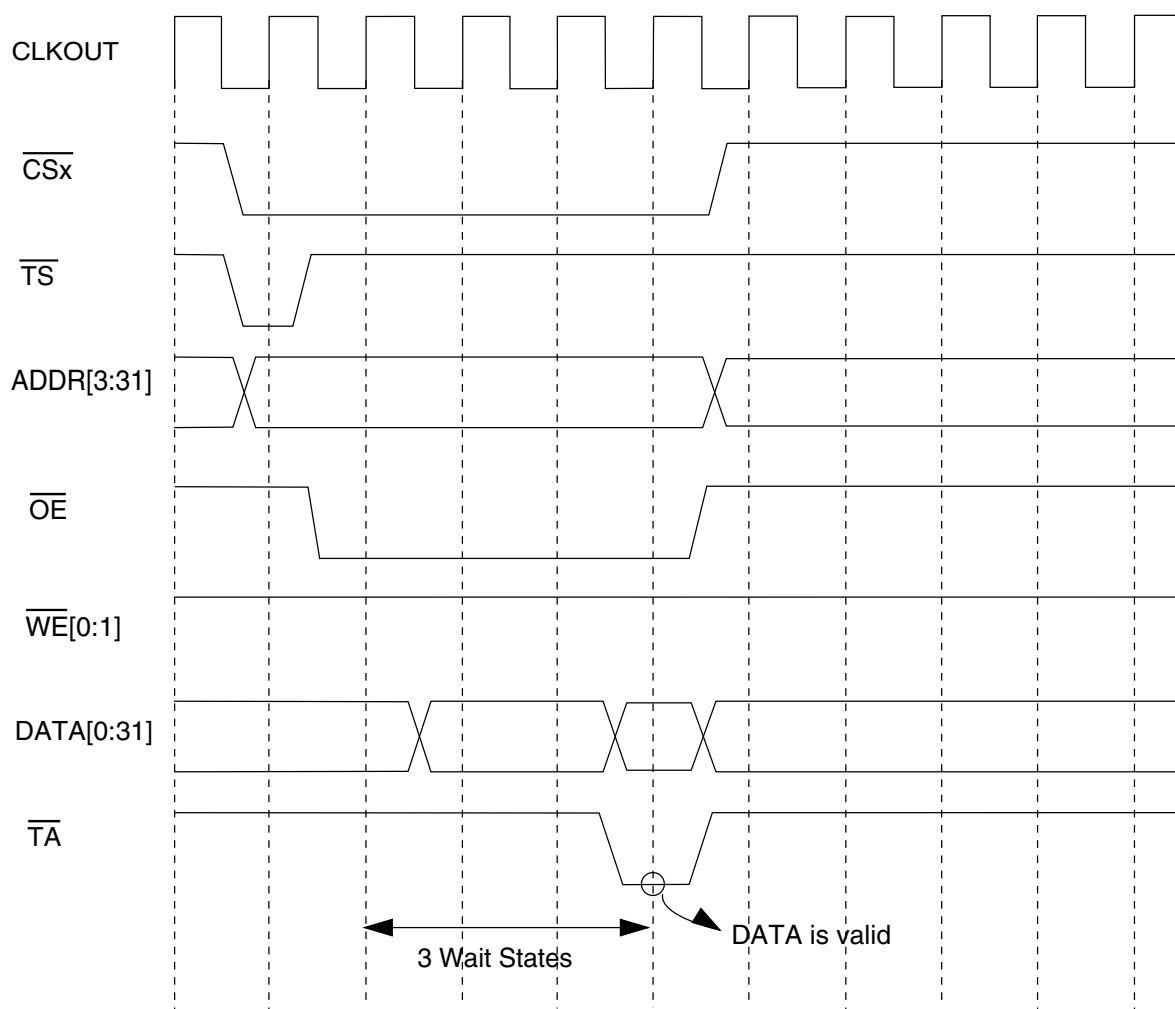


Figure 27-59. Read Operation to Asynchronous Memory, Three Initial Wait States

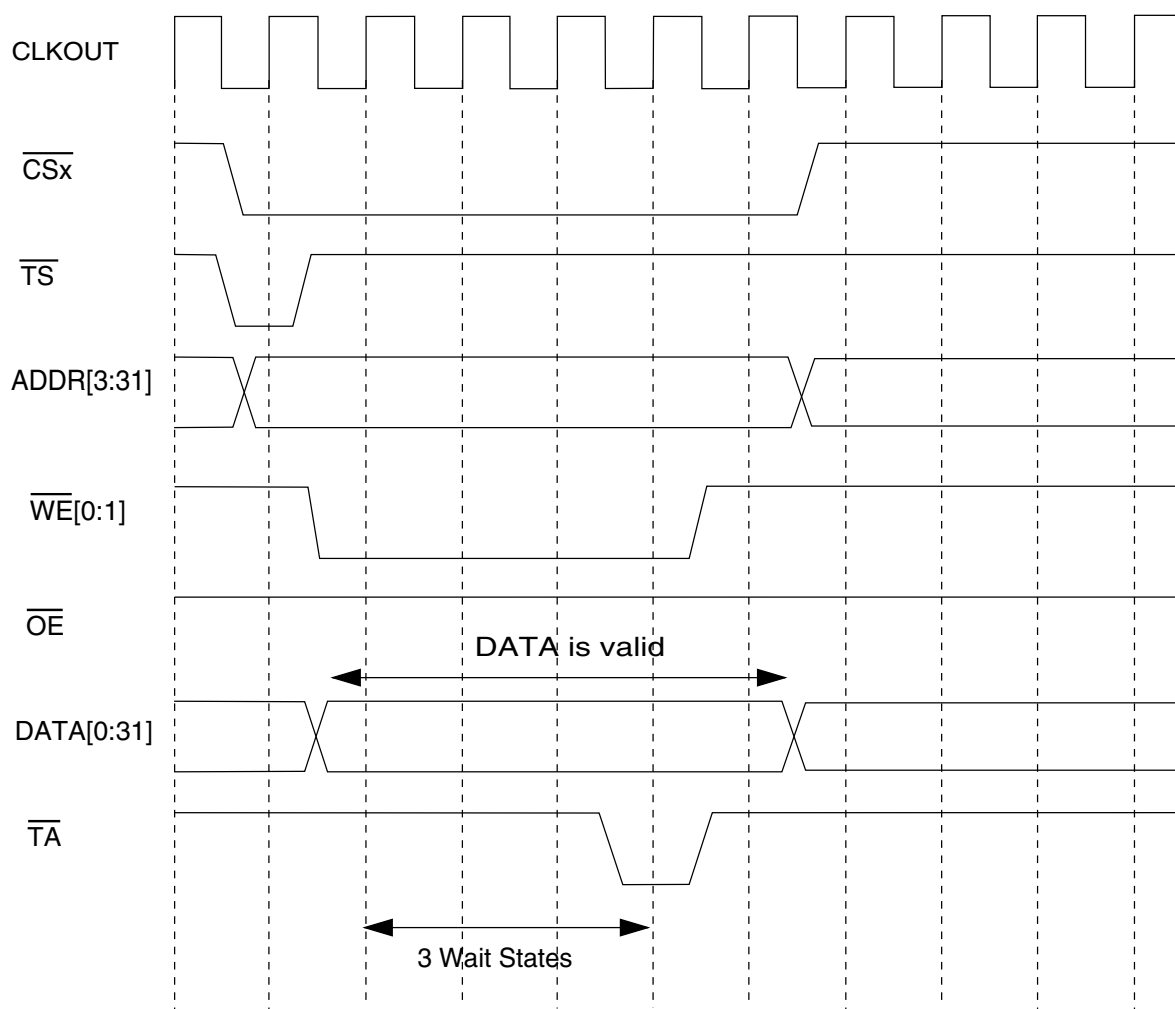
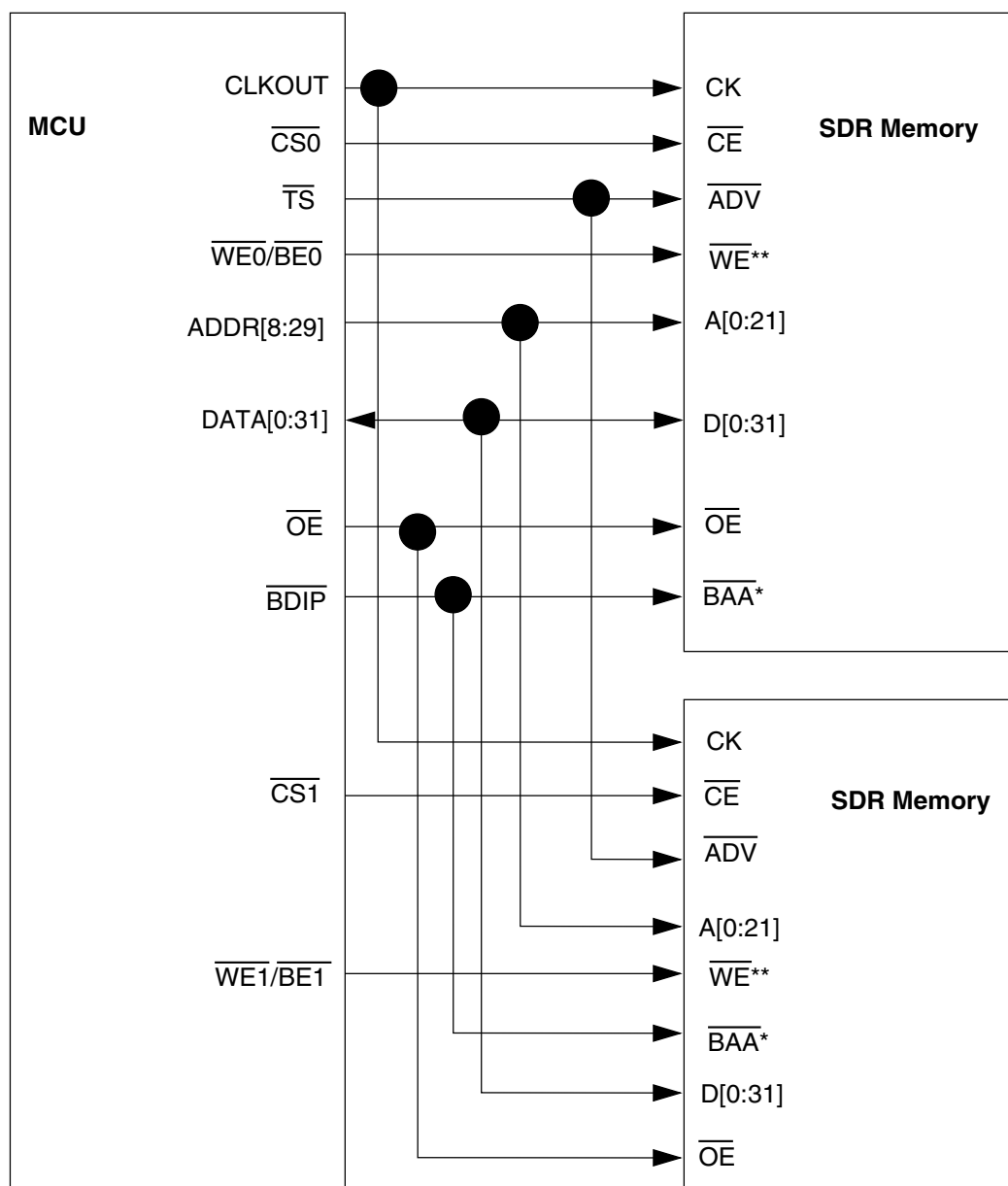


Figure 27-60. Write Operation to Asynchronous Memory, Three Initial Wait States

27.6.4 Connecting an MCU to Multiple Memories

The MCU can be connected to more than one memory at a time.

The following figure shows an example of how two memories could be connected to one MCU.



* May or may not be connected, depending on the memory used.

** Flash memories typically use one $\overline{WE}$ signal as shown, RAMs use 2 or 4 (16-bit or 32-bit)

Figure 27-61. MCU Connected to Multiple Memories

27.6.5 EBI Operation with Reduced Pinout MCUs

Some MCUs with this EBI may not have all the pins described in this document pinned out for a particular package. Some of the most common pins to be removed are DATA[16:31], arbitration pins ($\overline{BB}$, $\overline{BG}$, $\overline{BR}$), and TSIZ[0:1]. This section describes how

to configure dual-MCU systems for each of those scenarios, as well as describing limitations to EBI operation when other pins are missing ($\overline{\text{TA}}$, $\overline{\text{TEA}}$, $\overline{\text{BDIP}}$). More than one section may apply if the applicable pins are not present on one or both MCUs.

27.6.5.1 No Transfer Acknowledge ($\overline{\text{TA}}$) Pin

If an MCU has no $\overline{\text{TA}}$ pin available, this restricts the MCU to chip-select accesses only (no MCU->MCU transfers are possible). Non-chip-select accesses have no way for the EBI to know which cycle to latch the data. The EBI has no built-in protection to prevent non-chip-select accesses in this scenario; it is up to the user to make certain they set up chip-selects and external memories correctly to ensure all external accesses fall in a valid chip-select region.

27.6.5.2 No Transfer Error ($\overline{\text{TEA}}$) Pin

If an MCU has no $\overline{\text{TEA}}$ pin available, this eliminates the feature of terminating an access with $\overline{\text{TEA}}$. This means if an access times out in the EBI bus monitor, the EBI (master) will still terminate the access early, but there will be no external visibility of this termination, so the slave device might end up driving data much later, when a subsequent access is already underway. Therefore, the EBI bus monitor should be disabled when no $\overline{\text{TEA}}$ pin exists.

27.6.5.3 No Burst Data in Progress ($\overline{\text{BDIP}}$) Pin

If an MCU has no $\overline{\text{BDIP}}$ pin available, this eliminates burst support only if the burstable memory being used requires $\overline{\text{BDIP}}$ to burst. Many external memories use a self-timed configurable burst mechanism that does not require a dynamic burst indicator. Check the applicable external memory specification to see if $\overline{\text{BDIP}}$ is required in your system.

27.6.6 Summary of Differences from MPC5xx

Below is a summary list of the significant differences between this EBI and that of the MPC5xx parts.

- No memory controller support for external masters
 - must configure each master in multi-master system to drive its own chip selects
 - rationale: save complexity, no requirement for this feature

- Burst mechanism updated to be compatible with e200z core with 32-byte cache line
 - rationale: required for performance and compatibility with e200z core
- Removed these variable timing attributes from Option Register:
 - CSNT, ACS, TRLX, EHTR
 - rationale: reduces tester edgesets and complexity, no clear requirements for these features
- Removed reservation support on external bus
 - rationale: reservation not supported on internal bus, useless to support on external
- Removed Address Type (AT), Write-Protect (WP), and dual-mapping features
 - rationale: these functions can be replicated by Memory Management Unit (MMU) in e200z core
- Removed support for 8-bit ports
 - rationale: reduces complexity and not required
- Removed boot chip-select operation
 - rationale: on-chip Boot Assist Module (BAM) handles boot (and configuration of EBI registers)
- Open drain mode and pullup resistors no longer required for multi-master systems, extra cycle needed to switch between masters
 - rationale: saves customer hassle for multi-master system setup, at negligible performance cost
- Address decoding for external master accesses uses 4-bit code to determine internal slave instead of straight address decode
 - rationale: needed for compatibility with internal bridge address decoding and memory map
- Removed support for 3-master systems
 - rationale: very difficult to manage with internal bridge address decoding method and keep memory maps unique; not an essential feature to justify complexity of supporting
- Removed LBDIP Base Register bit, now late $\overline{\text{BDIP}}$ assertion is default behavior

- rationale: unaware of any memories that require $\overline{\text{BDIP}}$ to assert earlier than $\overline{\text{LBDIP}}$ timing, so reduce number of CS control bits and complexity
- Modified arbitration protocol to require extra cycles when switching between masters
 - rationale: could not use exact Oak protocol and make timing for full-speed operation; adding dead cycles to protocol allows bus to run full-speed in external master mode and makes this feature not limit overall EBI frequency
- Modified TSIZ[0:1] functionality to only indicate size of current transfer, not give information on ensuing transfers that may be part of the same atomic sequence
 - rationale: simpler and more intuitive functionality, no clear requirement for anything else
- Added support for 32-bit coherent read & write non-chip-select accesses in 16-bit data bus mode
 - rationale: some internal registers must be accessed all 32 bits at once to function as expected
- Added misaligned access support
 - rationale: some eSys cores require use of misaligned accesses for optimum performance
- Added calibration access support
 - rationale: support related device logic added to multiple eSys devices's, requested customer feature
- Added support for larger external address bus (up to 29 bits)
 - rationale: support larger external memory sizes
- Added support for address/data multiplexing
 - rationale: new feature to reduce minimum pin count
- Added support for using either half of data bus for 16-bit port transfers
 - rationale: helps A/D muxed usability, while maintaining backwards compatibility

Chapter 28

Analog Modules Overview

28.1 Introduction

This chip has the following modules for the primary purpose of analog measurement:

- Two EQADC modules, each containing two internal ADCs
- Four Sigma-Delta (SD) ADC modules
- Two Temperature Sensors
- Twelve Decimation Filters

28.2 EQADC subsystem

The integration of the two EQADC modules on this chip, EQADC_A and EQADC_B, is orthogonal except for the connection of the ADCs to specific external pins and internal channels.

Each EQADC module:

- has a hardware interface to twelve on-chip Decimation Filter modules to allow transfers of conversion and filtered values between the EQADC and Decimation Filters without CPU or DMA interaction. EQADC_A also has this hardware interface to the REACM2 module.
- has internal channel connections with the Power Management Controller module.

The following diagram shows the connections of the two EQADCs to other on-chip components and external pins.

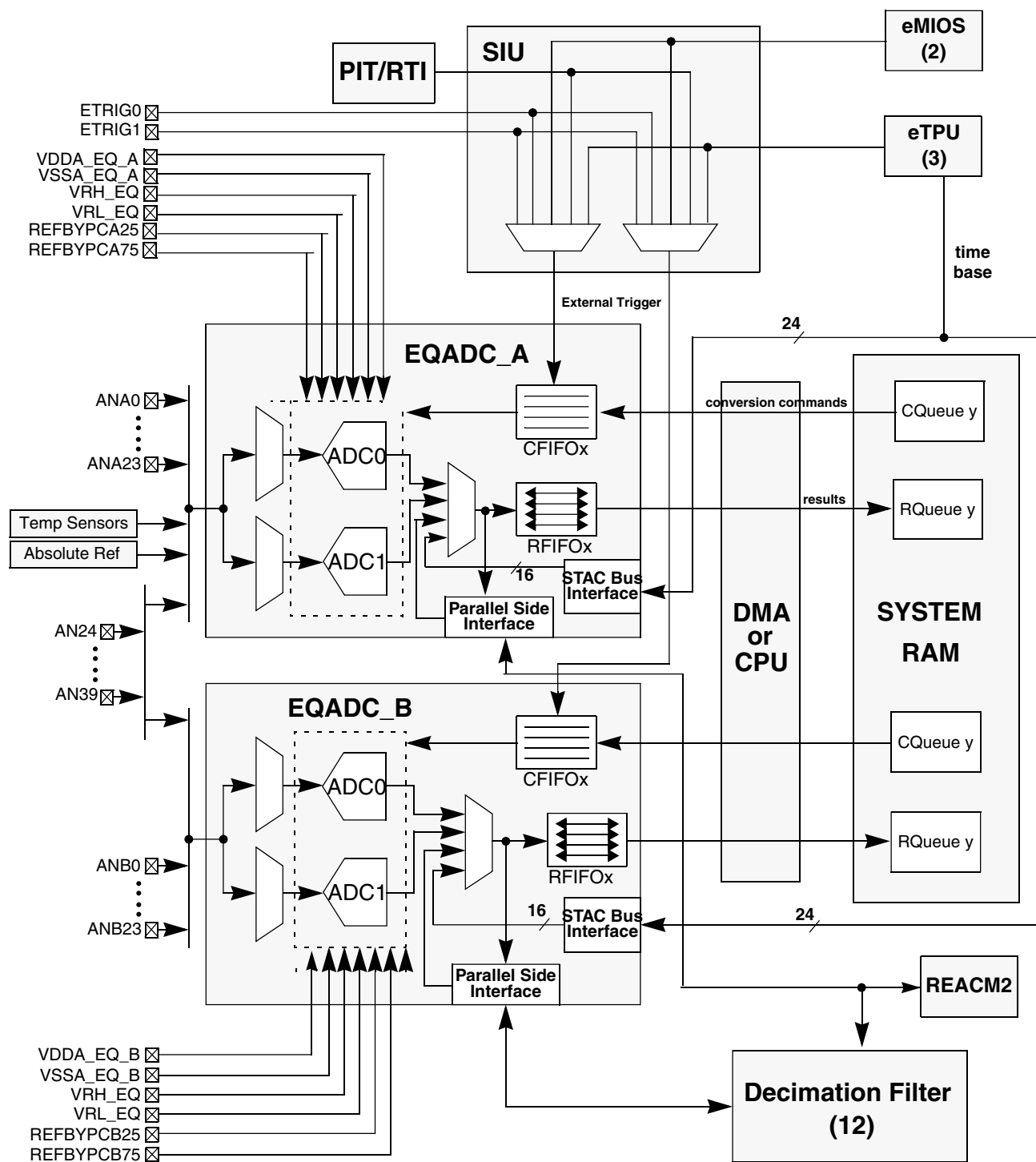


Figure 28-1. EQADC subsystem

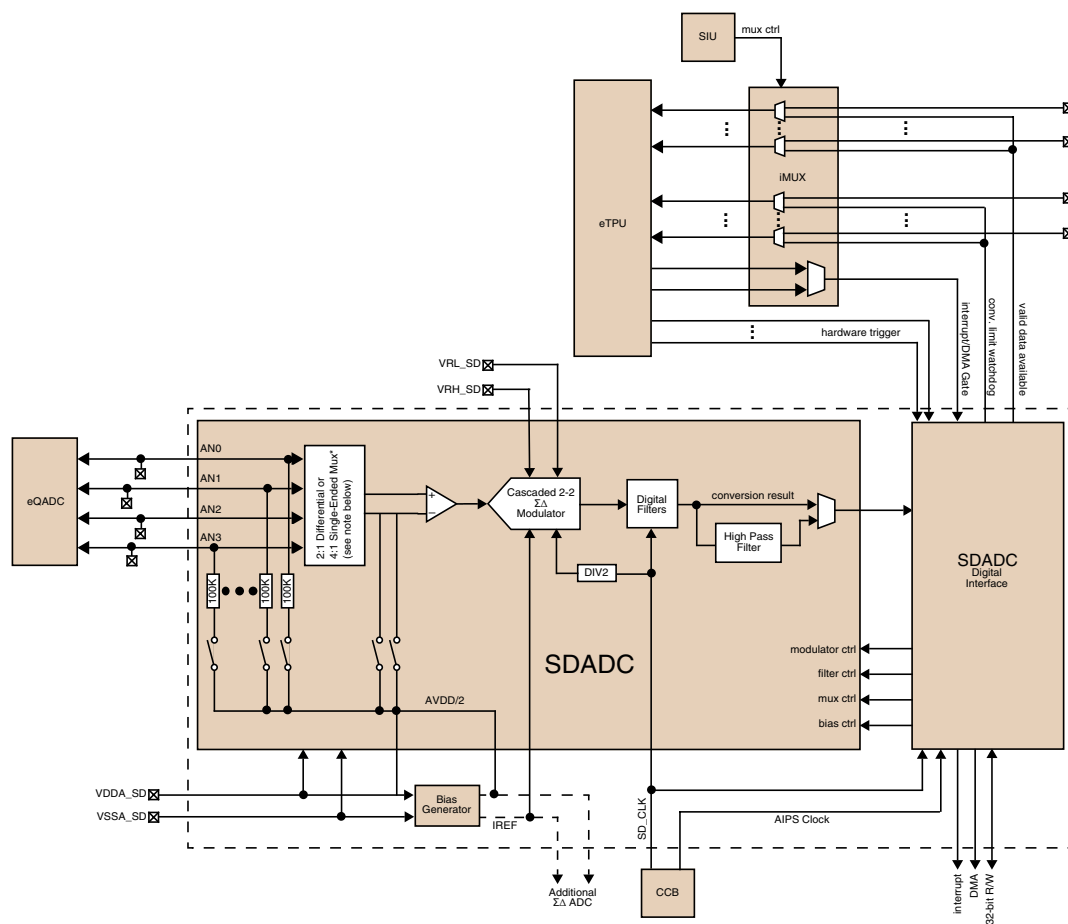
28.3 SDADC connections to other modules

The integration of the four SDADC modules on this chip is orthogonal except for the connection of the ADCs to specific external pins and internal channels.

Each SDADC module:

- has an independent digital interface with register interfaces, control hardware, DMA and interrupt requests, and so on
- connects to a subset of external pins that also connect to an eQADC module
- connects to eTPU channels to support:
 - conversion result interrupt and DMA requests and flag to eTPU (with hardware gating)
 - high/low conversion limit checking with interrupt request and flag to eTPU

The following diagram shows the connections of an SDADC module to other on-chip components and external pins.



NOTE: SDADC_4 has a 4:1 differential or 8:1 single-ended mux.

Figure 28-2. SDADC connections to other modules

Chapter 29

Sigma Delta Analog-to-Digital Converter (SDADC) Digital Interface

29.1 Chip-specific SDADC information

This chip has four instances of the SDADC module: SDADC_1, SDADC_2, and SDADC_3, and SDADC_4.

The following table summarizes varying features among the instances.

Table 29-1. SDADC instance channels and muxes

Instance	# channels	Channels	Single-ended input mux	Differential input mux
SDADC1, SDADC_2, and SDADC_3	4	AN[0] to AN[3]	4:1	2:1
SDADC_4	8	AN[0] to AN[7]	8:1	4:1

29.1.1 SDADC triggers and synchronous start

Each SDADC instance can be triggered to start a conversion either by a software write to [Software Trigger Key Register \(SDADC_STKR\)](#) or by a hardware trigger input selected in [Module Configuration Register \(SDADC_MCR\)](#) by writing the [TRIGSEL](#) field. See [Hardware triggering](#) for more information about triggering SDADC instances.

To support the synchronous start of all SDADC instances, the trigger output of each instance connects as a trigger input to every other instance.

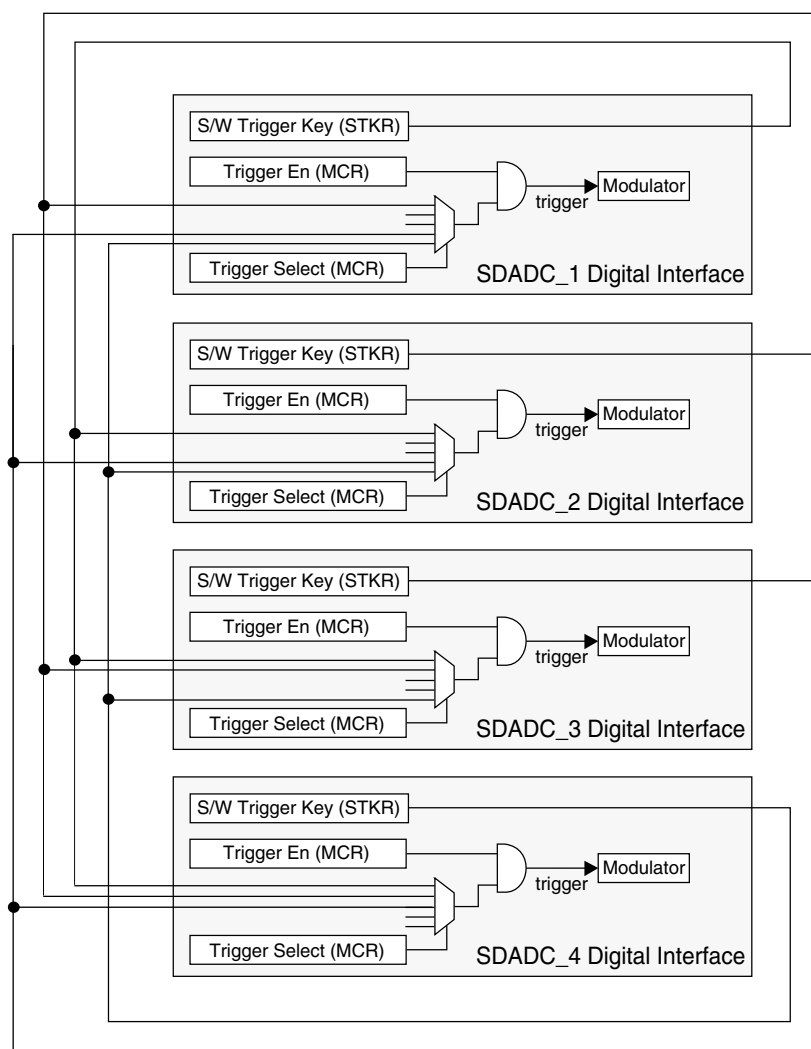


Figure 29-1. SDADC simultaneous start connectivity

Additional hardware trigger inputs connect to eTPU channels. The following table shows all of the hardware trigger inputs for each SDADC instance.

Table 29-2. SDADC trigger inputs

MCR[TRIGSEL] value	SDADC_1 trigger input	SDADC_2 trigger input	SDADC_3 trigger input	SDADC_4 trigger input
0000	Reserved	SDADC_1 trigger output	SDADC_1 trigger output	SDADC_1 trigger output
0001	SDADC_2 trigger output	Reserved	SDADC_2 trigger output	SDADC_2 trigger output
0010	SDADC_3 trigger output	SDADC_3 trigger output	Reserved	SDADC_3 trigger output
0011	SDADC_4 trigger output	SDADC_4 trigger output	SDADC_4 trigger output	Reserved
0100 - 0111	Reserved	Reserved	Reserved	Reserved
1000	eTPU_A channel 20	eTPU_A channel 22	eTPU_A channel 24	eTPU_A channel 26
1001	eTPU_A channel 21	eTPU_A channel 23	eTPU_A channel 25	eTPU_A channel 27
1010	eTPU_B channel 20	eTPU_B channel 22	eTPU_B channel 24	eTPU_B channel 26
1011	eTPU_B channel 21	eTPU_B channel 23	eTPU_B channel 25	eTPU_B channel 27

Table continues on the next page...

Table 29-2. SDADC trigger inputs (continued)

MCR[TRIGSEL] value	SDADC_1 trigger input	SDADC_2 trigger input	SDADC_3 trigger input	SDADC_4 trigger input
1100	eTPU_C channel 15	eTPU_C channel 17	eTPU_C channel 19	eTPU_C channel 21
1101	eTPU_C channel 16	eTPU_C channel 18	eTPU_C channel 20	eTPU_C channel 22
1110 - 1111	Reserved	Reserved	Reserved	Reserved

29.1.2 Connections among SDADCs and eTPUs

For information about SDADC and eTPU connections, see:

- [SDADC triggers and synchronous start](#)
- [SDADC connections to other modules](#)
- SIU details about the mapping of DMA/interrupt gating and mux advance inputs as well as the mapping of end-of-conversion and watchdog flag outputs

29.2 Introduction

The Sigma-Delta Analog-to-Digital Converter (SDADC) digital interface block controls the on-chip SDADC and holds control and status registers accessible for application. It provides accurate conversion data for a wide range of applications.

29.3 Overview

The SDADC digital interface block provides all control information for operating mode selection (single-ended or differential), analog input gain, decimation rate, high-pass filter enable, etc. It also generates the bias controls for common mode voltage selection, to bias the positive and negative terminals to different voltage levels (for example, half scale bias, reference ground, reference supply) for diagnostics. It generates the analog multiplexer control signals for the selected channels. To assist synchronous operation across multiple ADC blocks, hardware trigger support is implemented to start data conversion.

The following figure depicts the block diagram of an SDADC and its digital control block.

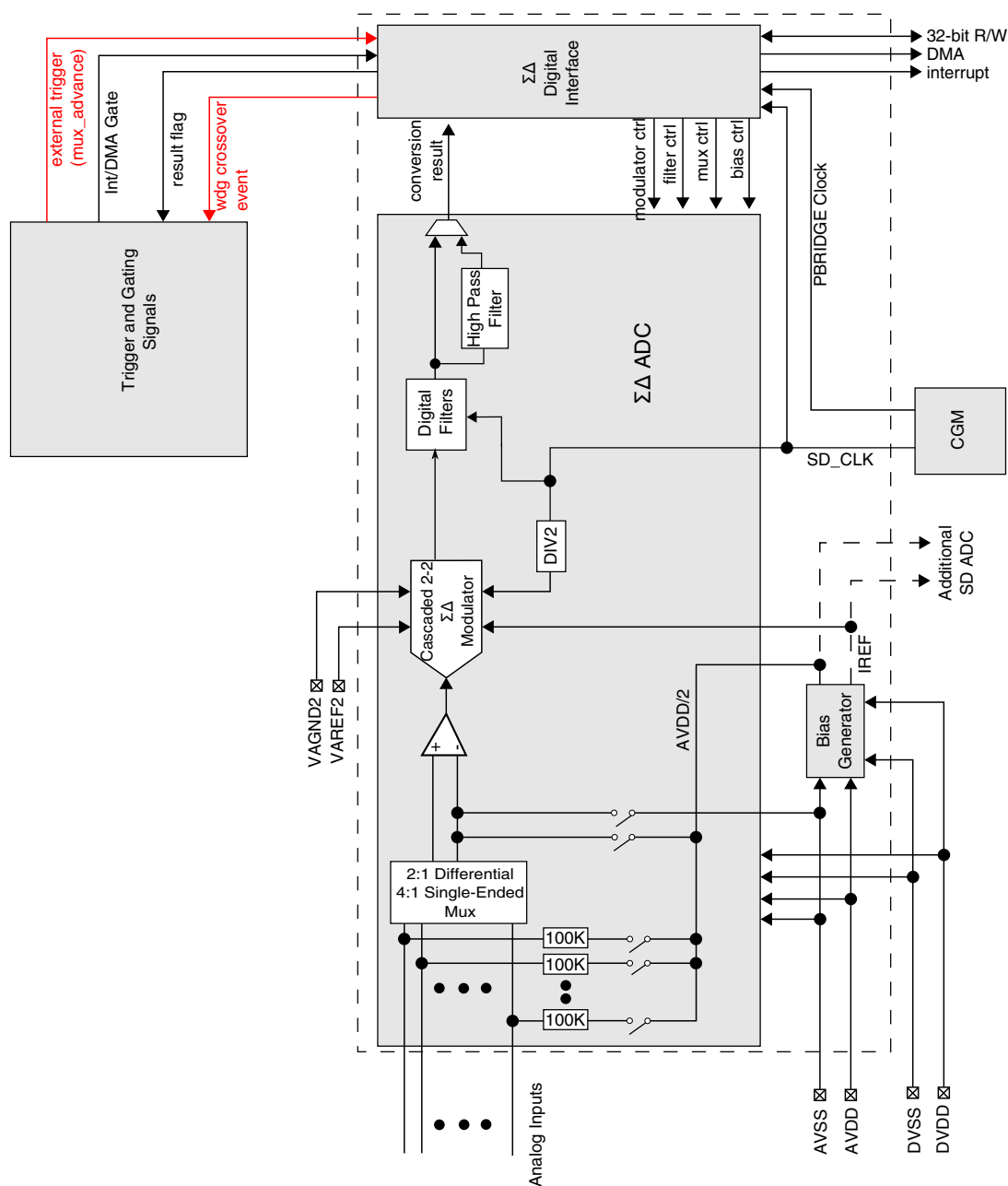


Figure 29-2. SDADC block diagram

29.4 Features

The features of the SDADC are as follows:

- 16-bit data resolution output
- Single-ended input or differential input mode of operation
- Programmable wraparound mechanism for both modes of operation

- Configurable trigger sources: hardware or software
- Configurable initial entry and wraparound values for the loop
- Configurable biasing for negative input terminal in single-ended mode
- Gain and offset calibration support using fixed bias for input terminals
- Programmable decimation rate
- Programmable gain for analog inputs
- Optional high-pass filter for pure AC application
- Hardware trigger support for synchronous operation of multiple SDADC blocks
- Trigger event output generation by software
- Programmable FIFO structure for storing up to 16 datawords of 16-bit converted data
- Interrupt/DMA request generation based on various conditions:
 - Data buffer above threshold
 - Data buffer overrun
 - Watchdog (WDG) crossover event
- Programmable threshold range watchdog support
- Low consumption mode
- Optional conversion process halt mechanism on SoC debug request

29.5 Modes of operation

This section describes the different operation modes of the SDADC. When exiting reset, the mode of operation is determined by the Module Configuration Register (MCR).

29.5.1 Differential input mode

This mode is entered by negating the MCR[MODE] bit. In this mode, a pair of analog inputs are connected to positive and negative terminals of ADC modulator. In order to support gain and offset calibration for diagnostics, it is possible in this mode to select fixed bias voltages to both input terminals.

29.5.2 Single-ended input mode

This mode is entered by asserting the MCR[MODE] bit. In this mode, the negative input terminal is biased with a fixed voltage based on selection information in MCR, and the other analog input is connected to the selected external analog channel.

29.6 External signal description

The following table shows the list of signals driven by external pins. See the chip-specific details about SDADC signals and package pinouts for information about signal muxing and availability external to the chip.

Table 29-3. Signal properties

Name	Function	I/O	Reset	Pull-up
AN[0:7]	Single-ended analog inputs or Differential pairs of analog inputs	I	—	—
VREFP	Voltage reference for SDADC	I	—	—
VREFN	Ground reference for SDADC	I	—	—
AVDD	Analog voltage supply for SDADC	I	—	—
AVSS	Analog ground supply for SDADC	I	—	—
DVDD	Digital voltage supply for SDADC	I	—	—
DVSS	Digital ground supply for SDADC	I	—	—

29.7 Memory map and register definition

This section provides the memory map for the SDADC block.

SDADC memory map

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
0	Module Configuration Register (SDADC_MCR)	32	R/W	0000_0000h	29.7.1/1057
4	Channel Selection Register (SDADC_CSR)	32	R/W	0000_0000h	29.7.2/1060
8	Reset Key Register (SDADC_RKR)	32	R/W	0000_A50Fh	29.7.3/1062
C	Status Flag Register (SDADC_SFR)	32	R/W	0000_0100h	29.7.4/1063
10	Request Select and Enable Register (SDADC_RSER)	32	R/W	0000_0000h	29.7.5/1065
14	Output Settling Delay Register (SDADC_OSDR)	32	R/W	0000_0000h	29.7.6/1067
18	FIFO Control Register (SDADC_FCR)	32	R/W	0000_0006h	29.7.7/1068
1C	Software Trigger Key Register (SDADC_STKR)	32	R/W	0000_0000h	29.7.8/1070
20	Converted Data Register (SDADC_CDR)	32	R	0000_0000h	29.7.9/1070
24	WDG Threshold Register (SDADC_WTHHLR)	32	R/W	0000_0000h	29.7.10/1071

29.7.1 Module Configuration Register (SDADC_MCR)

The Module Configuration register (MCR) consists of different control bits to select the operating modes and various configuration settings which define the behavior of the SDADC block.

Address: 0h base + 0h offset = 0h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0			PDR					0	PGAN			Reserved	Reserved	0	HPFEN
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	WDGEN	TRIGEDSEL		TRIGEN	TRIGSEL				FRZ	0		VCOMSEL	WRMODE	GECEN	MODE	EN
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

SDADC_MCR field descriptions

Field	Description																												
0–2 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.																												
3–7 PDR	Programmable Decimation Rate This field selects the over-sampling ratio to be applied to support different passbands with a fixed input sampling clock. The output data rate f_d is equal to $f_s / (2 \times \text{OSR})$, where f_s is the input sampling clock frequency. <table> <tr><td>00000</td><td>OSR = 24</td></tr> <tr><td>00001</td><td>OSR = 28</td></tr> <tr><td>00010</td><td>OSR = 32</td></tr> <tr><td>00011</td><td>OSR = 36</td></tr> <tr><td>00100</td><td>OSR = 40</td></tr> <tr><td>00101</td><td>OSR = 44</td></tr> <tr><td>00110</td><td>OSR = 48</td></tr> <tr><td>00111</td><td>OSR = 56</td></tr> <tr><td>01000</td><td>OSR = 64</td></tr> <tr><td>01001</td><td>OSR = 72</td></tr> <tr><td>01010</td><td>OSR = 75</td></tr> <tr><td>01011</td><td>OSR = 80</td></tr> <tr><td>01100</td><td>OSR = 88</td></tr> <tr><td>01101</td><td>OSR = 96</td></tr> </table>	00000	OSR = 24	00001	OSR = 28	00010	OSR = 32	00011	OSR = 36	00100	OSR = 40	00101	OSR = 44	00110	OSR = 48	00111	OSR = 56	01000	OSR = 64	01001	OSR = 72	01010	OSR = 75	01011	OSR = 80	01100	OSR = 88	01101	OSR = 96
00000	OSR = 24																												
00001	OSR = 28																												
00010	OSR = 32																												
00011	OSR = 36																												
00100	OSR = 40																												
00101	OSR = 44																												
00110	OSR = 48																												
00111	OSR = 56																												
01000	OSR = 64																												
01001	OSR = 72																												
01010	OSR = 75																												
01011	OSR = 80																												
01100	OSR = 88																												
01101	OSR = 96																												

Table continues on the next page...

SDADC_MCR field descriptions (continued)

Field	Description
	01110 OSR = 112 01111 OSR = 128 10000 OSR = 144 10001 OSR = 160 10010 OSR = 176 10011 OSR = 192 10100 OSR = 224 10101 OSR = 256 10110 - 11111 Reserved
8 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
9–11 PGAN	Programmable Gain This field selects the gain to be applied to the analog input stage of the SDADC. The effective analog input becomes the input voltage level multiplied by the gain factor. NOTE: When SDADC_MCR[PGAN]=111b (programmable gain is 16), the least significant bit of the conversion data in SDADC_CDR[CDATA] is always 0. 000 Gain = 1 001 Gain = 2 010 Gain = 4 011 Gain = 8 100 Reserved 101 Reserved 110 Reserved 111 Gain = 16
12 Reserved	This field is reserved. This field is reserved and always has the value 0. Only the reset value shall be written to this register field.
13 Reserved	This field is reserved. This field is reserved and always has the value 0. Only the reset value shall be written to this register field.
14 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
15 HPFEN	High Pass Filter Enable 0 High-pass (DC removal) filter is disabled 1 High-pass (DC removal) filter is enabled
16 WDGEN	Watchdog Enable This bit enables the WDG monitor. 0 WDG is disabled 1 WDG is enabled
17–18 TRIGEDSEL	Trigger Edge Selection 00 Falling edge of trigger input is selected 01 Rising edge of trigger input is selected 10 Both edges of trigger input are selected 11 Both edges of trigger input are selected

Table continues on the next page...

SDADC_MCR field descriptions (continued)

Field	Description
19 TRIGEN	Trigger Enable This field enables the hardware trigger input to initiate a fresh conversion. Upon receiving a trigger event, SDADC reset input is asserted and deasserted synchronously with respect to the peripheral clock. The reset pulse width is fixed to four peripheral clock cycles. The trigger event is ignored if the SDADC internal modulator is not enabled ($MCR[EN] = 0$). 0 Trigger input is disabled 1 Trigger input is enabled
20–23 TRIGSEL	Trigger Input Selection This field selects which input will be used for hardware triggered conversions. 0000 Input trigger 0 is selected. See chip-specific information for trigger source. 0001 Input trigger 1 is selected. See chip-specific information for trigger source. 0010 Input trigger 2 is selected. See chip-specific information for trigger source. 0011 Input trigger 3 is selected. See chip-specific information for trigger source. 0100 Input trigger 4 is selected. See chip-specific information for trigger source. 0101 Input trigger 5 is selected. See chip-specific information for trigger source. 0110 Input trigger 6 is selected. See chip-specific information for trigger source. 0111 Input trigger 7 is selected. See chip-specific information for trigger source. 1000 Input trigger 8 is selected. See chip-specific information for trigger source. 1001 Input trigger 9 is selected. See chip-specific information for trigger source. 1010 Input trigger 10 is selected. See chip-specific information for trigger source. 1011 Input trigger 11 is selected. See chip-specific information for trigger source. 1100 Input trigger 12 is selected. See chip-specific information for trigger source. 1101 Input trigger 13 is selected. See chip-specific information for trigger source. 1110 Input trigger 14 is selected. See chip-specific information for trigger source. 1111 Input trigger 15 is selected. See chip-specific information for trigger source.
24 FRZ	Freeze This field enables stopping the SDADC conversions at the end of the current channel conversion when the chip enters debug mode. When the chip enters debug mode, further conversions by the SDADC analog block will be stopped, and converted data output from the SDADC analog block are ignored. The ongoing channel conversion will be completed and the converted data is stored. When the chip exits debug mode, SDADC resumes the job that was left before halting the conversions. 0 Conversions are not stopped 1 Conversions are stopped
25–26 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
27 VCOMSEL	Common Voltage Bias Selection This field selects the common voltage bias for the negative input terminal of the SDADC during single-ended mode ($MODE = 1$). 0 Negative input terminal is biased with VREFN 1 Negative input terminal is biased with VREFP/2 (half-scale bias)
28 WRMODE	Wrap-Around Mode This bit selects the wraparound mechanism for conversion of programmed sequence of channels.

Table continues on the next page...

SDADC_MCR field descriptions (continued)

Field	Description
	0 Wraparound mechanism disabled (in this case the default software control mechanism will be enabled) 1 Wraparound mechanism enabled
29 GECEN	Accurate Gain Error Mode Enable This bit selects the accurate gain error mode, and must be set during gain error calibration as well as during conversions requiring accurate gain error mode. It can be kept '0' if gain error calibration mode is not needed. 0 Gain error calibration mode disabled 1 Gain error calibration mode enabled
30 MODE	Mode selection 0 Differential input mode selected 1 Single-ended input mode selected
31 EN	Enable for SDADC block 0 SDADC internal modulator placed in low consumption mode 1 SDADC internal modulator enabled

29.7.2 Channel Selection Register (SDADC_CSR)

The Channel Selection Register (CSR) selects analog inputs AN[0:7] to be connected to positive and negative terminals of the SDADC and controls the biasing of analog inputs to half-scale bias through 100 kOhm resistance.

Address: 0h base + 4h offset = 4h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0								BIASEN							
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0					ANCHSEL_WRAP			0					ANCHSEL		
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

SDADC_CSR field descriptions

Field	Description
0–7 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
8–15 BIASEN	Bias enable

Table continues on the next page...

SDADC_CSR field descriptions (continued)

Field	Description																																																								
	If BIASEN[i] = 1, analog input AN[i] (could be single-ended input or differential analog input) is connected to half-scale bias through a 100 kOhm resistor.																																																								
16–20 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.																																																								
21–23 ANCHSEL_WRAP	Analog Channel Selection Wraparound Value (ANCHSEL_WRAP) When in wraparound mode, this field indicates the maximum value of the wraparound counter, after which it will wrap around back to initial value 000. This value should be programmed before enabling wraparound mechanism (MCR[WRMODE] = 1).																																																								
24–28 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.																																																								
29–31 ANCHSEL	Analog Channel Selection(ANCHSEL) Based on single-ended or differential mode of operation (MCR[MODE]), common mode voltage selection (MCR[VCOMSEL]), this field defines the connectivity of analog inputs to either positive or negative polarity terminals of the SDADC according to the following table. Wraparound Mode: • This field also indicates the initial entry value for the first loop of the wraparound sequence from where the wraparound counter will start incrementing when wraparound mode is entered (when wraparound mechanism enabled (MCR[WRMODE] = 1)). • If this field is written any time during wraparound mode (when MCR[WRMODE] = 1), the present value of the wraparound counter will be updated with the value of ANCHSEL - the next trigger will act on the updated channel, and the next increment will be on the updated channel. Here, the value of ANCHSEL chosen should be < ANCHSEL_WRAP; otherwise it will be ignored by hardware. • Wraparound will always be from initial counter value 000. <table><tr><th>MODE</th><th>VCOMSEL</th><th>ANCHSEL</th><th>INP (positive terminal)</th><th>INM (negative terminal)</th></tr><tr><td rowspan="16">1</td><td rowspan="8">0</td><td>000</td><td>AN[0]</td><td>VREFN</td></tr><tr><td>001</td><td>AN[1]</td><td>VREFN</td></tr><tr><td>010</td><td>AN[2]</td><td>VREFN</td></tr><tr><td>011</td><td>AN[3]</td><td>VREFN</td></tr><tr><td>100</td><td>AN[4]</td><td>VREFN</td></tr><tr><td>101</td><td>AN[5]</td><td>VREFN</td></tr><tr><td>110</td><td>AN[6]</td><td>VREFN</td></tr><tr><td>111</td><td>AN[7]</td><td>VREFN</td></tr><tr><td rowspan="8">1</td><td>000</td><td>AN[0]</td><td>VREFP/2</td></tr><tr><td>001</td><td>AN[1]</td><td>VREFP/2</td></tr><tr><td>010</td><td>AN[2]</td><td>VREFP/2</td></tr><tr><td>011</td><td>AN[3]</td><td>VREFP/2</td></tr><tr><td>100</td><td>AN[4]</td><td>VREFP/2</td></tr><tr><td>101</td><td>AN[5]</td><td>VREFP/2</td></tr><tr><td>110</td><td>AN[6]</td><td>VREFP/2</td></tr><tr><td>111</td><td>AN[7]</td><td>VREFP/2</td></tr></table>	MODE	VCOMSEL	ANCHSEL	INP (positive terminal)	INM (negative terminal)	1	0	000	AN[0]	VREFN	001	AN[1]	VREFN	010	AN[2]	VREFN	011	AN[3]	VREFN	100	AN[4]	VREFN	101	AN[5]	VREFN	110	AN[6]	VREFN	111	AN[7]	VREFN	1	000	AN[0]	VREFP/2	001	AN[1]	VREFP/2	010	AN[2]	VREFP/2	011	AN[3]	VREFP/2	100	AN[4]	VREFP/2	101	AN[5]	VREFP/2	110	AN[6]	VREFP/2	111	AN[7]	VREFP/2
MODE	VCOMSEL	ANCHSEL	INP (positive terminal)	INM (negative terminal)																																																					
1	0	000	AN[0]	VREFN																																																					
		001	AN[1]	VREFN																																																					
		010	AN[2]	VREFN																																																					
		011	AN[3]	VREFN																																																					
		100	AN[4]	VREFN																																																					
		101	AN[5]	VREFN																																																					
		110	AN[6]	VREFN																																																					
		111	AN[7]	VREFN																																																					
	1	000	AN[0]	VREFP/2																																																					
		001	AN[1]	VREFP/2																																																					
		010	AN[2]	VREFP/2																																																					
		011	AN[3]	VREFP/2																																																					
		100	AN[4]	VREFP/2																																																					
		101	AN[5]	VREFP/2																																																					
		110	AN[6]	VREFP/2																																																					
		111	AN[7]	VREFP/2																																																					

Table continues on the next page...

SDADC_CSR field descriptions (continued)

Field	Description				
	MODE	VCOMSEL	ANCHSEL	INP (positive terminal)	INM (negative terminal)
	0	0/1	000	AN[0]	AN[1]
			001	AN[2]	AN[3]
			010	AN[4]	AN[5]
			011	AN[6]	AN[7]
			100	VREFN	VREFN
			101	VREFP/2	VREFP/2
			110	VREFP	VREFN
			111	VREFN	VREFP

29.7.3 Reset Key Register (SDADC_RKR)

The Reset Key Register (RKR) is the target for a predefined key write operation used to reset the SDADC to start a fresh conversion.

Address: 0h base + 8h offset = 8h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0																RESET_KEY															
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	1	0	0	1	0	1	0	0	0	0	1	1	1	1

SDADC_RKR field descriptions

Field	Description
0–15 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
16–31 RESET_KEY	Reset Key This field, when written with 5AF0h, is used to generate the reset for the SDADC block and to reload the internal counter with the start value programmed in OSDR. Any write access to this register which is different from the predefined key is ignored. Read access will always return the inverted key.

29.7.4 Status Flag Register (SDADC_SFR)

The Status Flag Register (SFR) contains status and flag bits. These bits are set by hardware and reflect the status of the SDADC and indicate the occurrence of events that may generate an interrupt or a DMA request. Software can clear some flags by writing '1' to those bits. Writing a '0' to a flag has no effect.

Address: 0h base + Ch offset = Ch

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0												ANCHSEL_CNT			
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	
R	0							DFEF	0				WTHH	WTHL	CDVF	DFORF	DFFF
W													w1c	w1c		w1c	w1c
Reset	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	

SDADC_SFR field descriptions

Field	Description
0–12 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
13–15 ANCHSEL_CNT	Analog Channel Selection Counter(ANCHSEL_CNT) This field reflects the current value of the internal Analog Channel Selection Counter. It indicates the present number of the channel selected through the analog multiplexer.
16–22 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
23 DFEF	Data FIFO Empty Flag This field is asserted when there is no dataword in the FIFO.

Table continues on the next page...

SDADC_SFR field descriptions (continued)

Field	Description
	0 Data FIFO is not empty 1 Data FIFO is empty
24–26 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
27 WTHH	Watchdog Upper Threshold Cross Over Event This field is set when Converted Data crosses Upper Threshold of WDG. It can be cleared by software writing '1' to clear. NOTE: This bit is also cleared by an acknowledgement from the DMA module when all of the following conditions are met: • SDADC_RSER[WTHDIRE]=1 (When this field is set by the occurrence of a watchdog upper threshold cross over event, a request is generated) • SDADC_SFR[WTHDIRS]=1 (The request generated when this field is set is a DMA request) • SDADC_RSER[GDIGE]=0 OR, if SDADC_RSER[GDIGE]=1, the external gating signal is asserted 0 Watchdog Upper Threshold Cross Over Event did not occur 1 Watchdog Upper Threshold Cross Over Event occurred
28 WTHL	Watchdog Lower Threshold Cross Over Event This field is set when Converted Data crosses Lower Threshold of WDG. It can be cleared by software writing '1' to clear. NOTE: This bit is also cleared by an acknowledgement from the DMA module when all of the following conditions are met: • SDADC_RSER[WTHDIRE]=1 (When this field is set by the occurrence of a watchdog lower threshold cross over event, a request is generated) • SDADC_SFR[WTHDIRS]=1 (The request generated when this field is set is a DMA request) • SDADC_RSER[GDIGE]=0 OR, if SDADC_RSER[GDIGE]=1, the external gating signal is asserted 0 Watchdog Lower Threshold Cross Over Event did not occur 1 Watchdog Lower Threshold Cross Over Event occurred
29 CDVF	Converted Data Valid Flag This field gives the actual status of data coming from the SDADC block. The converted data is not valid or is corrupted whenever there is a change in configuration settings determined by MCR, channel selection programmed in CSR. Any write access to the registers MCR and CSR will reset this flag. This flag is set automatically after an internal timer counts a number of output clock fd clock cycles specified by the value of SDADC_OSDR[OSD]. 0 Data output from SDADC is not valid 1 Data output from SDADC is valid
30 DFORF	Data FIFO Overrun Flag This field indicates that software or DMA failed to prevent FIFO from overflowing with converted datawords. Software can clear this bit by writing '1' to it after reading the FIFO contents through CDR. This flag remains set if software or DMA failed to clear the condition that caused this flag setting. If DFORF is set, further datawords will not be received into FIFO even if sufficient room exists. 0 No overrun has occurred since the last time the flag was cleared 1 Overrun has occurred or the DFORF has not been cleared since the last overrun occurred

Table continues on the next page...

SDADC_SFR field descriptions (continued)

Field	Description
31 DFFF	Data FIFO Full Flag This bit is set when the number of converted datawords in the FIFO is equal to or more than the number indicated by FCR[FTHL D]. Software can clear this bit by writing '1' to it. This bit is also cleared by an acknowledgement from the DMA controller when DMA request generation is selected. This bit remains set if software or DMA failed to clear the condition that caused this flag setting. Even if DFFF is set, datawords will continue to be received until an overrun condition occurs. Datawords will not be received if the converted data is not valid (CDVF = 0). NOTE: Whenever MCR[EN] is cleared in order to stop the SDADC or change the channel configuration, it is necessary to also clear the RSER[DFFDIRE] bit alongwith. Similarly if DMA or software is too slow in not reading the FIFO datawords resulting in DFFF condition it is advisable to stop the SDADC by clearing the MCR[EN] followed by clearing of RSER[DFFDIRE] bit. Both these action will ensure safe operation. 0 The number of datawords in FIFO is less than the number indicated by FCR[FTHL D] 1 The number of datawords in FIFO is equal to or greater than the number indicated by FCR[FTHL D] at some point in time since this flag was last cleared

29.7.5 Request Select and Enable Register (SDADC_RSER)

The Request Select and Enable Register (RSER) serves two purposes:

- Enables the flag bits in SFR to generate DMA requests or interrupt requests.
- Selects the type of request to generate.

Refer to the field descriptions for the type of requests that are supported.

Address: 0h base + 10h offset = 10h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	
R	0													WTHDIRS	DFFDIRS		
W																	
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	
R	GDIGE	0												WTHDIRE	CDVEE	DFORIE	DFFDIRE
W																	
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	

SDADC_RSER field descriptions

Field	Description
0–13 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
14 WTHDIRS	WDG Threshold Cross Over Event DMA/Interrupt Request Select When SFR[WTHH/WTHL] and RSER[WTHDIRE] fields are set, this field selects between generating a DMA request or an interrupt request. 0 Interrupt request is selected 1 DMA request is selected
15 DFFDIRS	Data FIFO Full DMA/Interrupt Request Select When the SFR[DFFF] and SFR[DFFDIRE] fields are set, this field selects between generating a DMA request or an interrupt request. 0 Interrupt request is selected 1 DMA request is selected
16 GDIGE	Global DMA/Interrupt Gating Enable This field determines whether all module interrupts and DMA requests need to be gated by an external gating signal or not. 0 No impact of external gating signal on module DMA/interrupt requests 1 All module DMA/interrupt requests are qualified only when external gating signal is asserted
17–27 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
28 WTHDIRE	WDG Threshold Cross Over Event DMA/Interrupt Request Enable This field enables the SFR[WTHH/WTHL] field to generate a request. The WTHDIRS field determines which is selected, a DMA request or an interrupt request. The final DMA/interrupt request also depends on the gating signal if enabled by the GDIGE field. 0 Interrupt/DMA request is disabled on WDG Threshold Cross Over Event 1 Interrupt/DMA request is enabled on WDG Threshold Cross Over Event
29 CDVEE	Conversion Data Valid Event Enable The status of SFR[CDVF] gated with RSER[CDVEE] is passed on as an event on “conversion_flag” output port, which can be used at SoC level for indication of valid data conversion window. Once enabled, the event output will remain asserted as long as valid data conversions are ongoing. 0 Event output disabled 1 Event output assertion/deassertion based on the SFR[CDVF] field
30 DFORIE	Data FIFO Overrun Interrupt Enable This bit enables the SFR[DFORF] field to generate an interrupt request. The final interrupt request also depends on the gating signal if enabled by the GDIGE field. 0 Interrupt request is disabled when data FIFO overrun condition occurs 1 Interrupt request is enabled when data FIFO overrun condition occurs
31 DFFDIRE	Data FIFO Full DMA/Interrupt Request Enable This field enables the SFR[DFFF] field to generate a request. The DFFDIRS field determines which is selected, a DMA request or an interrupt request. The final DMA/interrupt request also depends on the gating signal if enabled by the GDIGE field.

Table continues on the next page...

SDADC_RSER field descriptions (continued)

Field	Description
0	Interrupt/DMA request is disabled when data FIFO full condition occurs
1	Interrupt/DMA request is enabled when data FIFO full condition occurs

29.7.6 Output Settling Delay Register (SDADC_OSDR)

The Output Settling Delay Register (OSDR) provides a delay value to qualify the converted output data coming from the SDADC.

Address: 0h base + 14h offset = 14h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0																							OSD								
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	

SDADC_OSDR field descriptions

Field	Description
0–23 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
24–31 OSD	Output Settling Delay This field defines the delay to qualify the conversion data stored in CDR. Whenever the SDADC block is reset in order to start the conversion from a fresh state, an internal timer is loaded with this start value. The counter counts down with output clock f_d until it reaches zero, and then it generates a flag which qualifies the converted data. The output clock f_d is equal to $f_s / (2 \times \text{OSR})$, where f_s is the input sampling clock frequency. See the electrical specifications for the settling delays. NOTE: Refer to the t_{settling} and t_{LATENCY} parameters in the device datasheet for the minimum allowed output settling delay values.

29.7.7 FIFO Control Register (SDADC_FCR)

The FIFO Control Register (FCR) provides the ability to enable or disable the FIFO functionality, and to control the amount of available FIFO depth to be used.

NOTE

1. It is important to note that the FIFO depth is set at the factory and cannot be changed but the amount of available FIFO space actually used can be set by specifying the value in the FTHLD field. The FSIZE field indicates the amount of FIFO space available. By setting the value of FTHLD to be less than the FIFO depth indicated by FSIZE, the FIFO is considered to be full when the number of data words in the FIFO is equal or greater than FTHLD.
2. This register may be read at any time, but should only be written when SDADC is disabled (MCR[EN] = 0).

Address: 0h base + 18h offset = 18h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0															0
W																FRST
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0													FSIZE		
W					FTHLD								FOWEN			FE
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	0

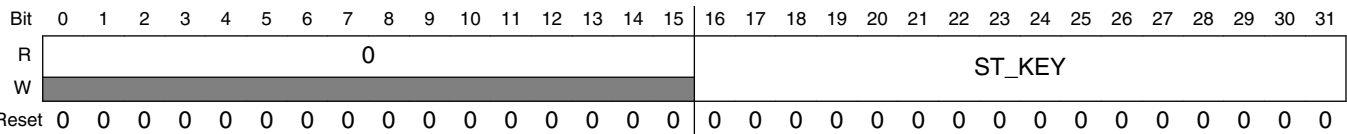
SDADC_FCR field descriptions

Field	Description
0–14 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
15 FRST	FIFO Flush Reset This is functional soft reset to flush the current FIFO content and initialize the internal FIFO pointers. This is W1S(write one shot) bit which can be written as '1' but always read as '0'. Use this option only after disabling the ADC i.e. MCR[EN]='0'. 0 No effect. 1 Generate a single cycle reset event to flush the FIFO.
16–19 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
20–23 FTHLD	FIFO Threshold When the number of datawords in the data FIFO is greater than the value in FTHLD field, the FIFO full event is flagged. An interrupt or a DMA request will be generated as determined by DFFDIRE and DFFDIRS fields of RSER. For proper operation, the value in the FTHLD field must be set to be less than or equal to the FIFO size as indicated by FSIZE.
24–27 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
28 FOWEN	FIFO Over Write Enable When set the built-in FIFO structure allows the overwriting of new converted data over older converted data after the FIFO has number of entries equal to the FSIZE. In this case overrun will not be seen. 0 Data FIFO OW option disabled. 1 Data FIFO OW option enabled.
29–30 FSIZE	FIFO Size The maximum number of converted datawords that can be stored in the data FIFO before an overrun occurs. This field is read-only. The reset value of this field depends on the implementation parameter to define the FIFO size. 00 FIFO depth = 1 dataword 01 FIFO depth = 4 datawords 10 FIFO depth = 8 datawords 11 FIFO depth = 16 datawords
31 FE	FIFO Enable When this field is set the built-in FIFO structure is enabled. The size of the FIFO structure is indicated by the FSIZE field. If this field is not set then the FIFO depth is set to one dataword regardless of the value in the FSIZE field. 0 Data FIFO is not enabled for multiple datawords; FIFO depth is one dataword 1 Data FIFO is enabled; FIFO depth is indicated by FSIZE

29.7.8 Software Trigger Key Register (SDADC_STKR)

The Software Trigger Key Register (STKR) allows generating the trigger event output by a software write operation.

Address: 0h base + 1Ch offset = 1Ch



SDADC_STKR field descriptions

Field	Description
0–15 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
16–31 ST_KEY	Software Trigger Key This bitfield, when written with FFFFh, is used to generate a trigger event output which can be used to trigger conversions of multiple SDADC blocks synchronously depending on the chip's implementation. Any write access to this register which is different from the predefined key is ignored. Read access will always return 0000h.

29.7.9 Converted Data Register (SDADC_CDR)

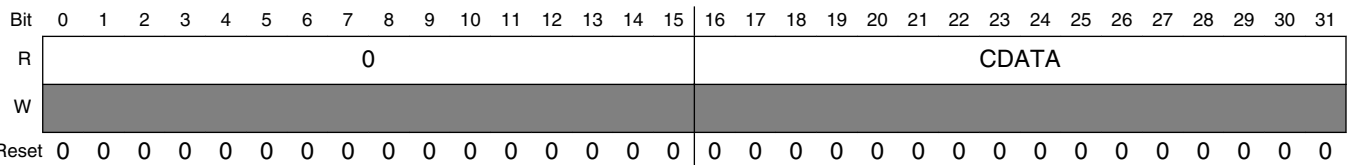
The Converted Data Register (CDR) stores the converted data in the data FIFO.

The conversion result in the CDATA word is a 16-bit, signed value.

NOTE

When SDADC_MCR[PGAN]=111b (programmable gain is 16), the least significant bit of the conversion data in SDADC_CDR[CDATA] is always 0.

Address: 0h base + 20h offset = 20h



SDADC_CDR field descriptions

Field	Description
0–15 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
16–31 CDATA	Converted Data Register (CDATA[13:0]) The converted datawords can be read from FIFO by reading this register. The data width is 16 bits which has actual 16-bit data (bits 15 down to 0) coming from ADC.

29.7.10 WDG Threshold Register (SDADC_WTHHLR)

This is the Converted Data Range WDG Threshold register.

NOTE

A single WDG monitor will check for lower and upper threshold crossover events.

Address: 0h base + 24h offset = 24h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	THRH															THRL																
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	

SDADC_WTHHLR field descriptions

Field	Description
0–15 THRH	WDG Upper Threshold Value This indicates the upper threshold for the converted data on any channel, above which threshold a crossover event would be flagged through SFR[WTHH].
16–31 THRL	WDG Lower Threshold Value This indicates the lower threshold for the converted data on any channel, below which threshold cross over event would be flagged through SFR[WTHL].

29.8 Functional description

The SDADC has the following distinct available modes determined by fields in MCR:

- Differential input mode
- Single-ended input mode
- Low consumption mode

29.8.1 Differential input mode

Differential input mode is entered when the SDADC_MCR[MODE] field is set to '0' and the SDADC_MCR[EN] field is set to '1'. In differential input mode, a pair of analog inputs among AN[0:7] is connected to the input terminals of the SDADC modulator block. The module supports four differential pairs: AN[0,1], AN[2,3], AN[4,5], and AN[6,7].

The modulator block has two input terminals, positive (INP) and negative (INM). The selection of input pair depends on the SDADC_CSR[ANCHSEL] field. The connectivity of each analog input in a given differential pair, whether it is connected to positive or negative terminal, is determined according to the Analog input AN[0:7] selection table in the Channel Selection Register description.

The differential input mode can also be used to perform gain and offset calibration, where the input terminals are biased with fixed voltages coming from the bias block.

29.8.2 Single-ended input mode

Single-ended input mode is entered by setting the SDADC_MCR[MODE] and SDADC_MCR[EN] fields to '1'. In this mode, one of the analog inputs among AN[0:7] is selected via the SDADC_CSR[ANCHSEL] field and is connected to the positive terminal. The negative input terminal is biased with a fixed internal voltage. There are two internal bias voltages available: reference ground (VREFN) and half-scale bias (VREFP/2). The selection of this common mode voltage is controlled via the SDADC_MCR[VCOMSEL] field.

29.8.3 Low consumption mode

This is the default mode after exiting from reset. This mode is exited by asserting the MCR[EN] field to enable the SDADC block. When data conversion is not required, the SDADC internal modulator can be put in low consumption mode to reduce power consumption.

29.8.4 Analog input multiplexing and bias support

The complete selection and connectivity of analog inputs AN[0:7] is based on the ANCHSEL field of CSR and the MODE and VCOMSEL bits of MCR, according to the Analog input AN[0:7] selection table in the Channel Selection Register description. Each analog input can be biased to half scale ($AVDD/2$) via on-chip 100 k Ω resistance. This allows simple AC coupling to the external analog inputs through a capacitor. This biasing is applicable irrespective of whether analog channel is selected through multiplexer or not.

29.8.5 Programmable gain and decimation rate

All analog inputs can be configured to have a selectable input gain as defined in the PGAN field description in MCR. This means the input signal is sampled and the result is amplified by the factor determined by PGAN before providing the same to modulator.

The SDADC module is provided with a fixed input sampling clock f_s . The input signal is grossly oversampled by the modulator in order to reduce quantization noise. To support different passbands with a fixed sampling clock, a programmable decimation rate is implemented. The process of decimation, to eliminate redundant data at the output while retaining the necessary information, is controlled by the field PDR in MCR.

29.8.6 High-pass filter support

For pure AC applications, it is useful to remove any DC component in the input signal. An optional high-pass filter is implemented which can be enabled by asserting the MCR[HPFEN] field. The -3 dB frequency of the filter is fixed (not programmable) and is specified in the device data sheet. This high-pass filter is implemented in the decimation filter logic of the SDADC.

29.8.7 Data conversion

The SDADC operates in a continuous conversion mode as soon as it is enabled. There is an inherent settling time required after SDADC startup before conversion data is considered valid.

NOTE

During that period, the Converted Data Valid Flag bit (SDADC_SFR[CDVF]) reads '0'.

Settling time duration depends on the latency and output settling time specifications. To account for the settling time, an internal timer is implemented that counts down from a start value determined by the value of the SDADC_OSDR[OSD] field.

The internal countdown timer is triggered by an SDADC reset, i.e., by writing the value 5AF0h to the SDADC_RKR[RESET_KEY] field.

NOTE

An SDADC reset is required any time there is a change to the SDADC configuration (via changes to the SDADC_MCR register settings), or a change to analog channel selection (via SDADC_CSR register settings).

When the timer reaches '0', the SDADC_SFR[CDVF] field is asserted and converted data is loaded into data FIFO on every rising edge of the output clock (f_d).

Any write access to the SDADC_MCR or SDADC_CSR register causes the SDADC_SFR[CDVF] flag to be cleared, indicating that data coming from SDADC is not valid. If the SDADC_SFR[CDVF] flag is '0', the following flags are blocked from becoming asserted:

- Data FIFO Full Flag (SDADC_SFR[DFFF])
- Data FIFO Overrun Flag (SDADC_SFR[DFORF])

If the SDADC_SFR[DFFF] bit is already set, the data FIFO is already full before SDADC_SFR[CDVF] deassertion.

Wraparound Control Mechanism: Here a wraparound logic includes all analog channels of the input multiplexer from Ain("000") to Ain(CSR[ANCHSEL_WRAP]) in both Single ended and Differential ended mode in wraparound sequence triggered by hardware or software source:

- MCR[WRMODE] bit would enable or disable the wraparound mechanism. Clearing the WRMODE bit automatically switches FSM back to Software control mode.
- Every valid trigger advances the wraparound counter to next channel to be converted.
- Wraparound Counter: An internal wraparound counter which points to analog channel to be converted takes the initial entry value for the first loop of execution from CSR[ANCHSEL] while entering wraparound mode. From next loop onwards the counter wraparounds to default initial value "000". Maximum value of the counter (wraparound value) should be programmed in CSR[ANCHSEL_WRAP] before entering wraparound mode. The definition of ANCHSEL and ANCHSEL_WRAP is provided in the Analog input AN[0:7] selection table in the Channel Selection Register description.
- Upon entering or during wraparound mode user should take care to program ANCHSEL value less than ANCHSEL_WRAP value; otherwise hardware would reject the new ANCHSEL value.

- Possible trigger sources:
 - SOURCE1 (Software Bit): writing to STKR.ST_KEY at any time during the wraparound mode (irrespective of MCR[TRIGEN or TRIGSEL] setting) generates a trigger pulse. Each such trigger initiates a fresh conversion and advances the wraparound counter by one.

NOTE

Care should be taken while programming the TRIGEN/TRIGSEL of each SDADC instance that no unintended simultaneous SW triggering of multiple instances happen.

- SOURCE2 (Hardware Trigger): Here external trigger output enabled by MCR[TRIGEN or TRIGEDSEL or TRIGSEL] before entering wraparound mode for selective or all SDADC instances provides trigger. Each such valid trigger initiates a fresh conversion and advances the wraparound counter by one. One of the hw_trig_in ports can be used to connect the external trigger output.
- Next channel number to be converted can be updated by software even in the middle of wraparound sequence execution. This can be done by rewriting CSR[ANCHSEL]. After CSR[ANCHSEL] is rewritten, the next trigger takes updated value and the wraparound counter would increment from the updated value.
- Current wraparound counter value can be read from the SFR[ANCHSEL_CNT] bitfield.

29.8.8 Hardware triggering

Software Control Mechanism: SDADC conversion can be triggered by various hardware events coming from either external pins or internal timers. The field TRIGSEL of MCR selects which hardware event is to be used for triggering conversions. This trigger event, which is synchronous to the peripheral clock, is enabled by asserting MCR[TRIGEN]. As soon as a trigger event is detected, reset input of SDADC block is asserted synchronously for four cycles. This will ensure that SDADC starts from a fresh state when reacting to the input trigger event. The SDADC internal modulator must be enabled before enabling the trigger event — otherwise the trigger event will be ignored.

If the same trigger event is provided to multiple SDADC digital interface modules and the respective enable controls (TRIGEN bit MCR) are asserted, synchronous operation of multiple SDADC blocks can be achieved with output results being updated at the same time. Once the reset input is deasserted, the first valid conversion output is stored in data FIFO after a configurable delay determined by OSD[OSD].

It is also possible to generate a software trigger event output by a write access of STKR with a predefined key. This event output can be used as a trigger to achieve synchronous software start of multiple SDADC blocks.

In case of a hardware trigger, the skew/jitter will be less than 100 ns. It will be more than one ADC clock cycle due to synchronization. In case of a software trigger, all SDADCs start synchronous sampling.

Wraparound Control Mechanism: In wraparound control mode, hardware and software triggers are equally valid. The hardware trigger source should be selected before enabling the wraparound mode.

29.8.9 Interrupt/DMA request support

The SDADC has one condition that generates interrupts only and another that generates interrupts or DMA request.

The DFFF field is asserted when the data FIFO full condition is detected, indicating that the number of converted datawords stored in data FIFO is equal to or greater than the value indicated by threshold determined by FCR[FTHLD]. The interrupt or DMA request is generated only when RSER[DFFDIRE] is asserted. RSER[DFFDIRS] selects whether a DMA request or an interrupt request is generated.

The data FIFO overrun flag (DFORF) indicates the FIFO overflow condition. Further converted datawords cannot be received and data is lost. The interrupt request is generated only when RSER[DFORIE] is asserted.

Apart from above, a WDG threshold crossover event can also generate an interrupt or DMA request.

If RSER[GDIGE] is set, all module interrupt/DMA requests are qualified only when an external global gating signal is asserted.

29.8.10 Gain calibration support

To perform gain calibration, the following sequence needs to be applied.

1. Select differential mode of operation by writing MCR[MODE] to '0'.
2. Enable gain error calibration mode by writing MCR[GECEN] to '1'.
3. Configure the mux selection ANCHSEL field of CSR to '110'. This configuration will apply VREFP on positive terminal and VREFN on negative terminal.
4. Disable the bias on all input analog channels by writing ENBIAS field of CSR to 00h.

5. Disable the high-pass filter by deasserting MCR[HPFEN].
6. Generate a reset event by writing 5AF0h to RESET_KEY of RKR.
7. Read the digital output (CDR[CDATA]) stored in FIFO after the output settling time. Expected output if there is no gain error is 0111_1001_1001_1001b, corresponding to full positive scale after attenuation inserted by the internal filter (1×0.95). The measurement should be repeated to reduce contribution of noise during the calibration process. D_p is the average value of attenuated positive full scale given by $D_p = \text{AVERAGE}(\text{CDR}[\text{CDATA}])$.
8. Change the mux selection ANCHSEL field of CSR to '111'. This configuration will apply VREFN on positive terminal and VREFP on negative terminal.
9. Generate a reset event by writing 5AF0h to RESET_KEY of RKR.
10. Read the digital output (CDR[CDATA]) stored in FIFO after the output settling time. Expected output if there is no gain error is 1000_0110_0110_0110b, corresponding to full negative scale after attenuation inserted by the internal filter (-1×0.95). The measurement should be repeated to reduce contribution of noise during calibration process. D_n is the average value of attenuated negative full scale given by $D_n = \text{AVERAGE}(\text{CDR}[\text{CDATA}])$.
11. The SDADC calibrated gain can be calculated as:

$$\text{Gain} = (D_p - D_n) / 2^{16}$$
12. The measured gain value can be used to nullify the gain errors in the digital output. For calibrated conversion, the data CDR[CDATA] provided by the SDADC should be normalized using the calculated gain: $\text{CDATA}_{\text{norm}}$ is given by $\text{CDATA}_{\text{norm}} = \text{CDR}[\text{CDATA}] / \text{Gain}$.

Note

- a. The gain error calibration should only be done with gain equal to '1' (in other words, PGAN = 000). Nevertheless, the calculated gain can be applied to normalized data also for conversion with GAIN > 1.
- b. MCR[GECEN] = 1 ensures the accurate gain error mode is enabled. This bit should be set both during calibration process and after calibration, for application data conversion. Application data conversion must then be normalized using the measured gain.
- c. The higher the number of full-scale conversion (D_p , D_n) done, the higher the rejection of noise during the calibration. The number of conversion is dependent on application noise. It is recommended to run at least 16 conversions when calculating an average value.

29.8.11 Offset calibration support

To perform offset calibration, the following sequence must be applied.

1. Select differential mode of operation by writing SDADC_MCR[MODE] to '0'.
2. Configure the SDADC_CSR[ANCHSEL] mux selection field to '100' or '101' as required.
 - a. SDADC_CSR[ANCHSEL] = '100' in case of data conversion after calibration in "single ended mode with negative input= $V_{SS\_HV\_ADR\_D}$ "
 - b. SDADC_CSR[ANCHSEL] = '101' in case of data conversion after calibration in "differential mode" and "single ended mode with negative input = $(V_{DD\_HV\_ADR\_D} - V_{SS\_HV\_ADR\_D}) / 2$ "
3. Disable the bias on all input analog channels by writing ENBIAS field of CSR to 00h.
4. Disable the high pass filter by deasserting SDADC_MCR[HPFEN].
5. Generate a reset event by writing 5AF0h to RESET_KEY of RKR.
6. Read the digital output stored in FIFO after the output settling time.
7. The measured offset can be used to nullify the offset error in the digital output.
Expected output is 00_0000_0000_0000b. The SDADC offset can be calculated as:
Offset = Expected Output – Actual Output

Note

The offset must be calculated for each SDADC_MCR[PGAN] field setting since it is expected to vary with gain configuration of SDADC.

29.9 Initialization information

To initialize the SDADC registers for data conversion, the following sequence is required.

1. Enable the SDADC by asserting MCR[EN].
2. Configure MCR to select the required mode, polarity, common mode voltage, input gain, and decimation rate.
3. Enable high-pass filter if required.
4. Select the required analog channel for data conversion. It is possible to select the bias for each channel for AC coupling applications.
5. Configure OSD delay according to SDADC startup time or latency from reset exit.
6. Generate a reset event by writing 5AF0h to RESET_KEY of RKR. Otherwise, if data conversion need to be triggered by a hardware event, assert MCR[TRIGEN].

7. DFFF interrupt or DMA request will be generated after the data FIFO has reached threshold to indicate that data can be transferred.

29.9.1 Data conversion step

To acquire a data from SDADC, the following sequence is required:

1. Enable the SDADC by asserting MCR[EN].
2. Configure MCR to select the required mode, polarity, common mode voltage, input gain, and decimation rate.
3. Enable high-pass filter is required.
4. Select the required analog channel for data conversion. It is possible to select the bias for each channel for AC coupling applications.
5. Configure OSD delay according to SDADC required startup time or latency from reset exit.
6. Generate a reset event writing 5AF0h in RESET_KEY field of RKR register.
7. Wait till FIFO empty flag DFEF of SFR register is clear.
8. Read data by CDATA of CDR register.

Repeat steps from 6 to 8 for new acquisitions.

NOTE

The time elapsed between reset event and Read data by CDATA field of CDR register are:

- Reset event <- -> Data Valid flag(CDVS) = Output settling Time delay (OSDR has to be set at least to 16) (Refer to the t_{settling} and t_{LATENCY} parameters in the device datasheet for the minimum allowed output settling delay values.).
- Data Valid flag <- -> Data FIFO is not empty = $0.5 * \text{CLK\_Out} + 3 * \text{CLK\_In}$
- Data FIFO is not empty <- -> read Data (safety point) = $2 * \text{CLK\_In}$, where $\text{CLK\_Out} = \text{CLK\_In} / (2 * \text{OSR})$

Chapter 30

Enhanced Queued Analog-to-Digital Converter (EQADC)

30.1 Chip-specific EQADC information

This chip has two instances of the EQADC module: EQADC_A and EQADC_B.

Each EQADC module:

- controls two independent Redundant Signed Digit (RSD) analog-to-digital (ADC) converter cores: ADC0 and ADC1.
- has an independent digital interface with register interfaces, control hardware, DMA and interrupt requests, and so on.

Chip-specific features of the EQADCs include:

- 24 dedicated external analog input pins to each EQADC module
- 16 pins shared between the two modules
- 6 additional pins share the dedicated EQADC_B channel inputs for greater pinout flexibility
- No support for external converters or an external device; the SSI interface is not present
- Addition of a Burst Trigger Generator, which is described in the SIU chapter
- Addition of bias/diagnostic pull up/down in the pad cell to support sharing ADC pins with SDADC inputs
- Targets up to 1.5M Sample/s (ADC_CLK=24 MHz) for differential conversions

The value of the external bypass between capacitor REFBYPC and VRL should be 100 nf.

NOTE

Due to legacy reasons, both ADC0 and ADC1 must be enabled for accurate conversion results.

NOTE

An 8 ms wait time from VDDA power up to enabling ADC is required to pre-charge the external 100 nf capacitor on REFBYPC pin. This time must be guaranteed by crystal startup time plus reset duration or user. The ADC internal bias generator circuit will start up after 10 μ s upon VRH/VRL and VDDA/VSSA power up and produces a stable/required bias current to the pre-charge circuit, but the current to other analog circuits are disabled until ADCs are enabled. As soon as the ADCs are enabled, the bias currents to all of analog circuits will be enabled.

30.1.1 EQADC connections to other modules

Each EQADC module:

- has a hardware interface to twelve on-chip Decimation Filter modules to allow transfers of conversion and filtered values between the EQADC and Decimation Filters without CPU or DMA interaction. EQADC_A also has this hardware interface to the REACM2 module.
- has internal channel connections with the Power Management Controller module.

30.1.1.1 On-chip Decimation Filters as DSP module

The EQADC transfers commands from multiple Command FIFOs (CFIFOs) to the on-chip ADCs. The multiple Result FIFOs (RFIFOs) can receive data from the on-chip ADCs or from the on-chip DSP module (Decimation Filters). Data from the on-chip ADCs can be routed to the side interface, processed by the Decimation Filters, and then routed back through the side interface to the RFIFOs.

30.1.1.2 Sources of software and external hardware triggers

The EQADC supports software and external hardware triggers (via package pins) from the PIT-RTI, eTPU, and eMIOS modules to initiate transfers of commands from the CFIFOs to the on-chip ADCs.

30.1.1.3 Routing conversion results to REACM2 and DECFILTER modules

To route the results of an EQADC conversion to the REACM2 module or a DECFILTER module, use one or more of the [Alternate Configuration 1-14 Control Registers \(ADC_ACR1-14\)](#).

1. Define an alternate configuration, including a destination for conversion results, by programming an ADC_ACRn register.
2. For the desired conversion, select that ADC_ACRn register.

The value of ADC_ACRn[DEST] determines one or more destinations for the conversion result.

- A DEST value that is greater than 0 selects one or more destinations on the EQADC's parallel side interface (PSI).
- The REACM2 and DECFILTER modules are destinations on the PSI.

The DAM field of the [EQADC Module Configuration Register \(EQADC_MCR\)](#) also affects the destination selection.

See [Table 30-121](#) for general information about destinations selected by EQADC_MCR[DAM] and ADC_ACRn[DEST], including when DEST is 0. For DEST values greater than 0, that table shows the PSI destinations as numbered simply 1 to 15. The following table correlates those numbered destinations with the REACM2 and DECFILTER modules.

Table 30-1. Conversion destination selection details for PSI

EQADC instance	EQADC_MCR[DAM]	ADC_ACRn[DEST]	First destination	Second destination
EQADC_A and EQADC_B	0	0001	1: DECFILTER_A	—
		0010	2: DECFILTER_B	—
		0011	3: DECFILTER_C	—
		0100	4: DECFILTER_D	—
		0101	5: DECFILTER_E	—
		0110	6: DECFILTER_F	—
		0111	7: DECFILTER_G	—
		1000	1: DECFILTER_A	5: DECFILTER_E
		1001	2: DECFILTER_B	
		1010	3: DECFILTER_C	
		1011	4: DECFILTER_D	
		1100	1: DECFILTER_A	7: DECFILTER_G
		1101	2: DECFILTER_B	
		1110	3: DECFILTER_C	
		1111	4: DECFILTER_D	
	1	0001	1: DECFILTER_A	Depends on ADC_EACRn[DEST2] (Table 30-125)
		0010	2: DECFILTER_B	
		0011	3: DECFILTER_C	
		0100	4: DECFILTER_D	
		0101	5: DECFILTER_E	
		0110	6: DECFILTER_F	
		0111	7: DECFILTER_G	
		1000	8: DECFILTER_H	
		1001	9: DECFILTER_I	
		1010	10: DECFILTER_J	
		1011	11: DECFILTER_K	
		1100	12: DECFILTER_L	
		1101	—	—
EQADC_A		1110	14: REACM2	Depends on ADC_EACRn[DEST2] (Table 30-125)
EQADC_B		1110	—	—
EQADC_A and EQADC_B		1111	—	—

30.1.1.4 STAC server IDs

STAC server IDs are defined in [STAC server IDs](#) in the chip-specific eTPU information.

30.1.2 EQADC Stop mode entry

On this chip, SIUL_HLT1[EQADCx] is the chip-level control for issuing a request to an EQADC instance to enter its Stop mode.

30.1.3 EQADC pre-gain control

On this chip, the analog ADC option to select x1, x2, or x4 pre-gain is available only for differential conversions. Single-ended conversions can use only x1 pre-gain.

The PRE_GAIN field of the [Alternate Configuration 1-14 Control Registers \(ADC_ACR1-14\)](#) controls the selection.

30.1.4 CFIFO and RFIFO depth

Every CFIFO and RFIFO, including CFIFO0, has four entries.

30.1.5 EQADC pin pullup/pulldown control

This following table shows the analog and digital EQADC pins with pullup/pulldown control and the registers that provide the control.

Table 30-2. EQADC pins with pullup/pulldown and control registers

Pin function	EQADC_A	EQADC_B	Pullup/pulldown control registers
Analog	ANA0 to ANA7 ANA16 to ANA23	ANB0 to ANB7	ADC Pull Up/Down Control Register x (ADC_PUDCRx, x=0-39)
Digital	—	ANB8 to ANB23 ANB40 to ANB45	Pad Configuration Register (SIU_PCRn)

30.1.6 EQADC pin-to-channel mapping

For both EQADC_A and EQADC_B, the following table shows the pin-to-channel mapping to show which signals go to which ADC.

Table 30-3. EQADC_A and EQADC_B pin-to-channel mapping

Analog input pin on 416 MAPBGA package	Function	ADC number		EQADC_A channel number	EQADC_B channel number
		EQADC_A	EQADC_B		
ANA0-ANA23 ¹	Single ended conversion	0, 1	—	0-23	—
ANB0-ANB23	Single ended conversion	—	0, 1	—	0-23
AN24-AN39	Single ended conversion	0, 1	0, 1	24-39	24-39
—	VRH	0, 1	0, 1	40	40
—	VRL	0, 1	0, 1	41	41
—	50% (VRH-VRL) ²	0, 1	0, 1	42	42
—	75% (VRH-VRL)	0, 1	0, 1	43	43
—	25% (VRH-VRL)	0, 1	0, 1	44	44
—	Local bandgap 1220 mV	0, 1	—	45	—
ANA8	ANWA ³	0, 1	—	64-71	—
ANB8	ANWB	—	0, 1	—	64-71
ANA9	ANXA	0, 1	—	72-79	—
ANB9	ANXB	—	0, 1	—	23-79
ANA10	ANYA	0, 1	—	80-87	—
ANB10	ANYB	—	0, 1	—	80-87
ANA11	ANZA	0, 1	—	88-95	—
ANB11	ANZB	—	0, 1	—	88-95
ANA0 and ANA1	DANA0+ and DANA0-	0, 1	—	96	—
ANB0 and ANB1	DANB0+ and DANB0-	—	0, 1	—	96
ANA2 and ANA3	DANA1+ and DANA1-	0, 1	—	97	—
ANB2 and ANB3	DANB1+ and DANB1-	—	0, 1	—	97
ANA4 and ANA5	DANA2+ and DANA2-	0, 1	—	98	—
ANB4 and ANB5	DANB2+ and DANB2-	—	0, 1	—	98
ANA6 and ANA7	DANA3+ and DANA3-	0, 1	—	99	—
ANB6 and ANB7	DANB3+ and DANB3-	—	0, 1	—	99
—	Temperature Sensor 0	0, 1	—	128	—
—	Temperature Sensor 1	0, 1	—	129	—
—	PMC bandgap 1200 mV	0	—	145 ⁴	—
—	Core/1.2 V LVD cold sense	0	—	145 ⁴	—
—	Core/1.2 V LVD hot sense	0	—	145 ⁴	—
—	Core/1.2 V LVD hot reference	0	—	145 ⁴	—
—	Flash/3.3 V LVD divider tap	0	—	145 ⁴	—
—	Flash/3.3 V LVD reference (PMC bandgap 1200 mV)	0	—	145 ⁴	—
—	PMC/5 V LVD reference (PMC bandgap 1200 mV)	0	—	145 ⁴	—
ANB40	Single ended conversion	—	0, 1	—	128
ANB41	Single ended conversion	—	0, 1	—	129
ANB42	Single ended conversion	—	0, 1	—	130

Table continues on the next page...

Table 30-3. EQADC_A and EQADC_B pin-to-channel mapping (continued)

Analog input pin on 416 MAPBGA package	Function	ADC number		EQADC_A channel number	EQADC_B channel number
		EQADC_A	EQADC_B		
ANB43	Single ended conversion	—	0, 1	—	131
ANB44	Single ended conversion	—	0, 1	—	132
ANB45	Single ended conversion	—	0, 1	—	133
ANA16	ANRA	0, 1	—	224-231	—
ANB16	ANRB	—	0, 1	—	245-231
ANA17	ANSA	0, 1	—	232-239	—
ANB17	ANSB	—	0, 1	—	223-239
ANA18	ANTA	0, 1	—	240-247	—
ANB18	ANTB	—	0, 1	—	240-247
ANA19	ANUA	0, 1	—	248-255	—
ANB19	ANUB	—	0, 1	—	248-255

1. AN8x-ANx11 and ANx16-ANx19 have alternate functions when used with an external multiplexer.
2. After calibration, the 50% reference point reads approximately 20 mV below 50% (VRH-VRL). The 50% reference point should not be used to calibrate the EQADC. Use the 25% and 75% reference points to calibrate the EQADC.
3. This function is present when using an external multiplexer.
4. Select the desired PMC voltage using the channel select field [ADC_CS](#) of the [ADC Channel Select Register \(PMC_ADC_CS\)](#)

30.2 Introduction

30.2.1 Module overview

The Enhanced Queued Analog-to-Digital Converter (EQADC) block provides accurate and fast conversions for a wide range of applications. The EQADC provides a parallel interface to two on-chip analog-to-digital converters (ADCs), and a parallel side interface to one or more on-chip digital signal processing (DSP) modules (for example, a decimation filter). The two on-chip ADCs are architected to allow access to all the analog channels.

The EQADC transfers commands from multiple Command FIFOs (CFIFOs) to the on-chip ADCs. The multiple Result FIFOs (RFIFOs) can receive data from the on-chip ADCs or from an on-chip DSP module. Data from the on-chip ADCs can be routed to the side interface, processed by the on-chip DSP and then routed back through the side interface to the RFIFOs. The EQADC supports software and external hardware triggers from other blocks to initiate transfers of commands from the CFIFOs to the on-chip

ADCs. It also monitors the fullness of CFIFOs and RFIFOs, and accordingly generates DMA or interrupt requests to control data movement between the FIFOs and the system memory, which is external to the EQADC.

30.2.2 Block diagram

Figure 30-1 is the block diagram for the EQADC.

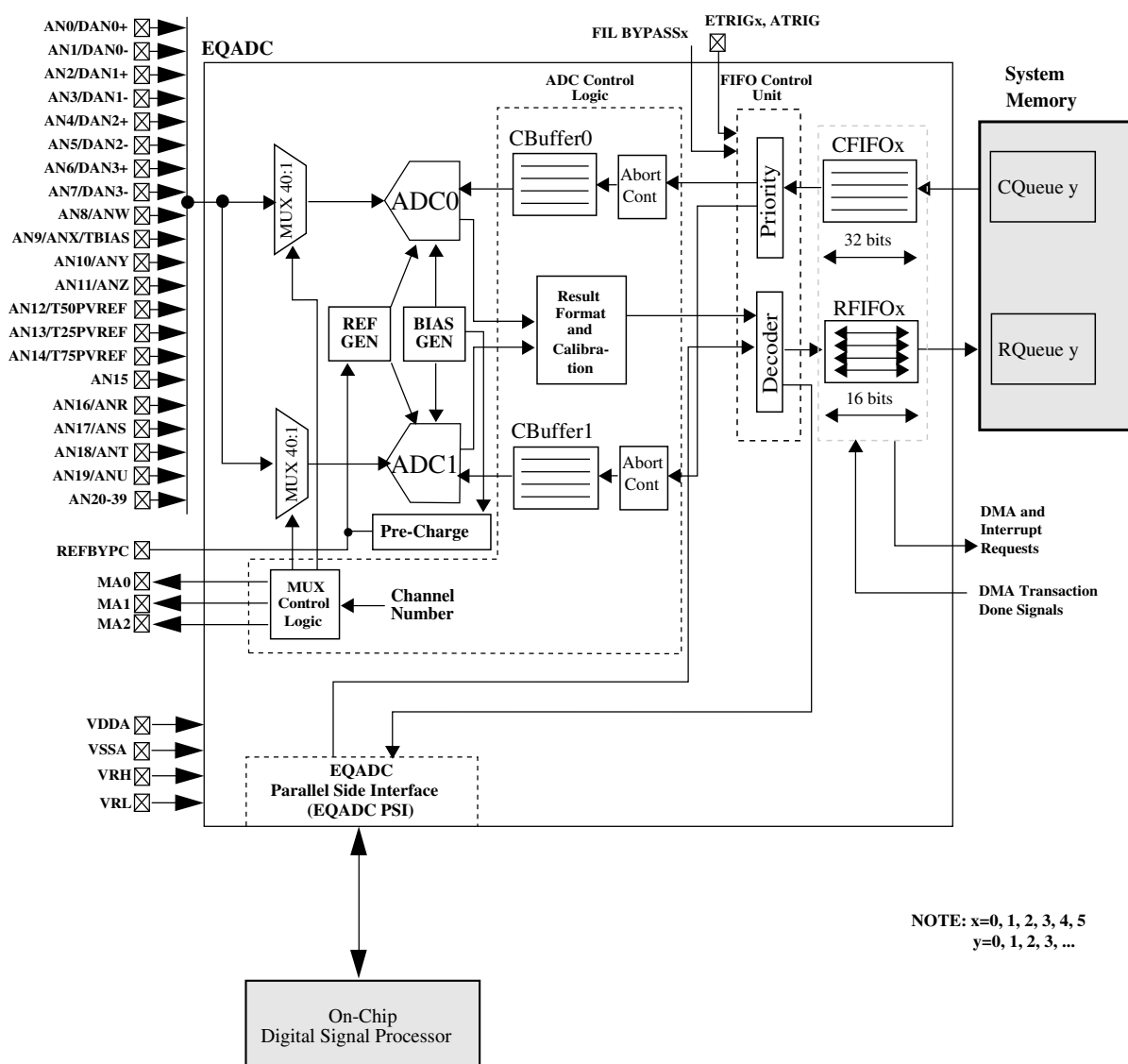


Figure 30-1. EQADC Block Diagram

Figure 30-1 shows the primary components inside the EQADC. The EQADC consists of the *FIFO Control Unit* which controls the CFIFOs and the RFIFOs, the *ADC Control Logic* which controls the two on-chip ADCs, and the *EQADC Parallel Side Interface*

(EQADC PSI) which allows communication with on-chip eQADC companion¹ modules. There are 6 CFIFOs and 6 RFIFOs, each with 4 entries, except CFIFO0 that can have 8 entries.

The *FIFO Control Unit* performs the following functions:

- It prioritizes the CFIFOs to determine what CFIFOs will have their commands transferred.
- Supports software and hardware triggers to start command transfers from a particular CFIFO.
- Decodes command data from the CFIFOs, and accordingly, sends these commands to one of the two on-chip ADCs.
- Decodes result data from on-chip ADCs and transfers data to the appropriate RFIFO or to the parallel side interface.

The *ADC Control Logic* manages the execution of commands bound for on-chip ADCs. It interfaces with the CFIFOs via two 2-entry command buffers (CBuffers) with abort control and with the RFIFOs and side interface via the *Result Format and Calibration Sub-Block*. The *ADC Control Logic* performs the following functions:

- Buffers command data for execution.
- Decodes command data and accordingly generates control signals for the two on-chip ADCs.
- Detects abort request, stores aborted commands and buffers immediate conversion commands.
- Formats and calibrates conversion result data coming from the on-chip ADCs.
- Generates the internal multiplexer control signals and the select signals used by the external multiplexers.

The EQADC PSI allows for a full duplex, synchronous, parallel communication between the EQADC and several on-chip companion modules.

Figure 30-1 also depicts data flow through the EQADC. Commands are contained in system memory in a user defined data structure. The most likely data structure to be used is a queue as depicted in the Figure 30-1². Command data is moved from the command

1. Depending on the device, companion modules may be DSP modules, decimation filters, or any other processing block.
 2. Command and result data can be stored in system memory in any user defined data structure. However, in this document it will be assumed that the data structure of choice is a queue, since it is the most likely data structure to be used and because queues are the only type of data structure supported by the DMAC.

queue (CQueue) to the CFIFOs by either the host CPU or by the DMAC. Once a CFIFO is triggered and becomes the highest priority CFIFO using a certain CBuffer, command data is transferred from the CFIFO to the on-chip ADCs. The ADC executes the command, and the result is moved through the *Result Format and Calibration Sub-Block* to either the side interface or to the RFIFO. Data from the on-chip companion module bypasses the *Result Format and Calibration Sub-Block* and is moved directly to its specified RFIFO. When data is stored in an RFIFO, data is moved from the RFIFO by the host CPU or by the DMAC to a data structure in system memory depicted in the [Figure 30-1](#) as a result queue (RQueue).

For users familiar with the QADC, the EQADC system upgrades the functionality provided by that block. See [EQADC versus QADC](#) for a comparison between the EQADC and QADC.

30.2.3 Features

The EQADC block includes these distinctive features:

- Two independent on-chip RSD Cyclic ADCs
 - 8, 10, and 12 bits Resolution
 - Differential conversions
 - Differential channels include variable gain amplifier for improved dynamic range (x1; x2; x4)
 - Differential channels include programmable pull-up and pull-down resistors for biasing and sensor diagnostics (200k ohms; 100k ohms; 5k ohms)
 - Sample times of 2 (default), 8, 64 or 128 ADC clock cycles
 - Provides time stamp information when requested
 - Parallel interface to EQADC CFIFOs and RFIFOs
 - Supports both right-justified unsigned and signed formats for conversion results
 - The REFBYPC pin stabilizes one of internal generated reference
 - Temperature sensor
 - Ability to measure directly Vdd
- Automatic application of ADC calibration constants

- Provision of reference voltages (25% VREF and 75% VREF) for ADC calibration purposes
- 40 input channels available to the two on-chip ADCs
- 4 pairs of differential analog input channels
- Parallel Side Interface to communicate with several on-chip companion modules
- STAC bus to import an alternative timebase to the internal time stamp
- Priority Based CFIFOs
 - Supports six CFIFOs with fixed priority. The lower the CFIFO number, the higher its priority. When commands of distinct CFIFOs are bound for the same CBuffer, the higher priority CFIFO is always served first.
 - Immediate conversion command feature with conversion abort control
 - Streaming mode operation of CFIFO0 to execute some commands several times
 - Supports software and several hardware trigger modes to arm a particular CFIFO
 - Generates interrupt when command coherency is not achieved
- External Hardware Triggers
 - Supports rising edge, falling edge, high level and low level triggers
 - Supports configurable digital filter
 - Supports controls to bypass the trigger digital filters
- Two Triggers operation mode for queue0
 - Additional internal trigger (not filtered) called Advance trigger that is used to enable the external trigger of queue0 and to control the loop behavior of CFIFO0
- Supports 4 to 8 external 8-to-1 muxes which can expand the input channel number from 40 to 96

30.3 Modes of operation

This section describes the operation modes of the EQADC.

30.3.1 Normal Mode

This is the default operational mode when the EQADC is not in background debug or stop mode.

30.3.2 Debug Mode

Upon a debug mode entry request, EQADC behavior will vary according to the status of the DBG field in [EQADC Module Configuration Register \(EQADC_MCR\)](#). If DBG is programmed to 0b00, the debug mode entry request is ignored. If DBG is programmed to 0b10 or to 0b11, the EQADC will enter debug mode.

During debug mode, the EQADC will not transfer commands from any CFIFOs, no null messages will be transmitted to the device, no data will be returned to any RFIFO, no hardware trigger event will be captured, and all EQADC registers can be accessed as in Normal mode. The latter implies that CFIFOs can still be triggered using software triggers, since no scheme is implemented to write-protect registers during debug mode. DMA and interrupt requests continue to be generated as in Normal Mode.

If at the time the debug mode entry request is detected, there are commands in the on-chip CBuffers that were already under execution, these commands will be completed but the generated results, if any, will not be sent to the RFIFOs until debug mode is exited. Commands whose execution has not started will not be executed until debug mode is exited.

The clock associated with an on-chip ADC stops, during its low phase, after the ADC ceases executing commands. The time base counter will only stop after all on-chip ADCs cease executing commands.

When exiting debug mode, the EQADC relies on the CFIFO operation modes and on the CFIFO status to determine the next command entry to transfer.

The EQADC internal behavior after the debug mode entry request is detected differs depending on the status of command transfers.

- No command transfer is in progress.

The EQADC immediately halts future command transfers from any CFIFO.

If a null message is being transmitted, EQADC will complete the serial transmission before halting future command transfers. If valid data (conversion result or data read from an ADC register) is received at the end of transmission, it will not be sent to an RFIFO until debug mode is exited.

If the null message transmission is aborted, the EQADC will complete the abort procedure before halting future command transfers from any CFIFO. The message of the CFIFO that caused the abort of the previous serial transmission will only be transmitted after debug mode is exited.

- Command transfer is in progress.

EQADC will complete the transfer and update CFIFO status before halting future command transfers from any CFIFO. Command transfers to the internal CBuffers are considered completed when a command is written to the buffers.

- Command/Null message transfer through serial interface was aborted but next serial transmission did not start.

If the debug mode entry request is detected between the time a previous serial transmission was aborted and the start of the next transmission, the EQADC will complete the abort procedure before halting future command transfers from any CFIFO. The message of the CFIFO that caused the abort of the previous serial transmission will only be transmitted after debug mode is exited.

30.3.3 Stop Mode

Upon a stop mode entry request detection, the EQADC progressively halts its operations until it reaches a static, stable state from which it can recover when returning to Normal mode. EQADC then asserts an acknowledge signal, indicating that it is static and that the clock input can be stopped. . The latter implies that, as long as the system clock is running, CFIFOs can still be triggered using software triggers, since no scheme is implemented to write-protect registers during stop mode.

If at the time the stop mode entry request is detected, there are commands in the on-chip CBuffers that were already under execution, these commands will be completed but the generated results, if any, will not be sent to the RFIFOs until stop mode is exited. Commands whose execution has not started will not be executed until stop mode is exited.

After these remaining commands are executed, the clock input to the ADCs and the bias generator circuit is stopped. The ADC clock stops during its low phase. The time base counter will only stop after all on-chip ADCs cease executing commands. Only then, the stop acknowledge signal is asserted. When exiting stop mode, the EQADC relies on the CFIFO operation modes and on the CFIFO status to determine the next command entry to transfer.

The EQADC internal behavior after the stop mode entry request is detected differs depending on the status of the command transfer.

- No command transfer is in progress

The EQADC immediately halts future command transfers from any CFIFO.

If a null message is being transmitted, EQADC will complete the transmission before halting future command transfers. If valid data (conversion result or data read from an ADC register) is received at the end of the transmission, it will not be sent to an RFIFO until stop mode is exited.

If the null message transmission is aborted, the EQADC will complete the abort procedure before halting future command transfers from any CFIFO. The message of the CFIFO that caused the abort of the previous serial transmission will only be transmitted after stop mode is exited.

- Command transfer is in progress

EQADC will complete the transfer and update CFIFO status before halting future command transfers from any CFIFO. Command transfers to the internal CBuffers are considered completed when a command is written to the buffers.

30.4 External signal description

30.4.1 Overview

The following is a list of external pins.

Note

At chip integration level, some of the digital and analog signals listed here might share pins or not be available to the chip. Refer to the Signals chapter for details.

Table 30-4. External signals

Name	Port	Function	Reset state	Type
AN0/DAN0+	Input	Single-ended analog input / Differential analog input positive terminal	—	Analog
AN1/DAN0-	Input	Single-ended analog input / Differential analog input negative terminal	—	Analog

Table continues on the next page...

Table 30-4. External signals (continued)

Name	Port	Function	Reset state	Type
AN2/DAN1+	Input	Single-ended analog input / Differential analog input positive terminal	—	Analog
AN3/DAN1-	Input	Single-ended analog input / Differential analog input negative terminal	—	Analog
AN4/DAN2+	Input	Single-ended analog input / Differential analog input positive terminal	—	Analog
AN5/DAN2-	Input	Single-ended analog input / Differential analog input negative terminal	—	Analog
AN6/DAN3+	Input	Single-ended analog input / Differential analog input positive terminal	—	Analog
AN7/DAN3-	Input	Single-ended analog input / Differential analog input negative terminal	—	Analog
AN8/ANW	Input	Single-ended analog input / Single-ended analog input from external multiplexers	—	Analog
AN9/ANX	Input	Single-ended analog input / Single-ended analog input from external multiplexers	—	Analog
AN10/ANY	Input	Single-ended analog input/Single-ended analog input from external multiplexers	—	Analog
AN11/ANZ	Input	Single-ended analog input / Single-ended analog input from external multiplexers	—	Analog
AN12	Input	Single-ended analog input	—	Analog
AN13	Input	Single-ended analog input	—	Analog
AN14	Input	Single-ended analog input	—	Analog
AN15	Input	Single-ended analog input	—	Analog
AN16/ANR ¹	Input	Single-ended analog input / Single-ended analog input from external multiplexers	—	Analog
AN17/ANS ¹	Input	Single-ended analog input / Single-ended analog input from external multiplexers	—	Analog
AN18/ANT ¹	Input	Single-ended analog input / Single-ended analog input from external multiplexers	—	Analog
AN19/ANU ¹	Input	Single-ended analog input / Single-ended analog input from external multiplexers	—	Analog
AN20	Input	Single-ended analog input	—	Analog
AN21	Input	Single-ended analog input	—	Analog
AN22	Input	Single-ended analog input	—	Analog
AN23	Input	Single-ended analog input	—	Analog
AN24	Input	Single-ended analog input	—	Analog
AN25	Input	Single-ended analog input	—	Analog
AN26	Input	Single-ended analog input	—	Analog
AN27	Input	Single-ended analog input	—	Analog
AN28	Input	Single-ended analog input	—	Analog
AN29	Input	Single-ended analog input	—	Analog
AN30	Input	Single-ended analog input	—	Analog

Table continues on the next page...

Table 30-4. External signals (continued)

Name	Port	Function	Reset state	Type
AN31	Input	Single-ended analog input	—	Analog
AN32	Input	Single-ended analog input	—	Analog
AN33	Input	Single-ended analog input	—	Analog
AN34	Input	Single-ended analog input	—	Analog
AN35	Input	Single-ended analog input	—	Analog
AN36	Input	Single-ended analog input	—	Analog
AN37	Input	Single-ended analog input	—	Analog
AN38	Input	Single-ended analog input	—	Analog
AN39	Input	Single-ended analog input	—	Analog
MA0	Output	External multiplexer control signal	0	Digital
MA1	Output	External multiplexer control signal	0	Digital
MA2	Output	External multiplexer control signal	0	Digital
VDDA	Input	Analog Positive Power Supply	—	Power
VSSA	Input	Analog Negative Power Supply	—	Power
VRH	Input	Voltage Reference High	—	Power
VRL	Input	Voltage Reference Low	—	Power
REFBYPC	Input	External Bypass capacitor Pin	—	Power
ETRIG0	Input	External trigger for CFIFO0	—	Digital
ETRIG1	Input	External trigger for CFIFO1	—	Digital
ETRIG2	Input	External trigger for CFIFO2	—	Digital
ETRIG3	Input	External trigger for CFIFO3	—	Digital
ETRIG4	Input	External trigger for CFIFO4	—	Digital
ETRIG5	Input	External trigger for CFIFO5	—	Digital

1. Can be disabled or not using configuration parameters.

30.4.2 Detailed signal descriptions

30.4.2.1 AN0/DAN0+ — Single-ended analog input/Differential analog input positive terminal

AN0 is a single-ended analog input to the two on-chip ADCs. DAN0+ is the positive terminal of the differential analog input DAN0 (DAN0+ - DAN0-).

30.4.2.2 AN1/DAN0- — Single-ended analog input/Differential analog input negative terminal

AN1 is a single-ended analog input to the two on-chip ADCs. DAN0- is the negative terminal of the differential analog input DAN0 (DAN0+ - DAN0-).

30.4.2.3 AN2/DAN1+ — Single-ended analog input/Differential analog input positive terminal

AN2 is a single-ended analog input to the two on-chip ADCs. DAN1+ is the positive terminal of the differential analog input DAN1 (DAN1+ - DAN1-).

30.4.2.4 AN3/DAN1- — Single-ended analog input/Differential analog input negative terminal

AN3 is a single-ended analog input to the two on-chip ADCs. DAN1- is the negative terminal of the differential analog input DAN1 (DAN1+ - DAN1-).

30.4.2.5 AN4/DAN2+ — Single-ended analog input/Differential analog input positive terminal

AN4 is a single-ended analog input to the two on-chip ADCs. DAN2+ is the positive terminal of the differential analog input DAN2 (DAN2+ - DAN2-).

30.4.2.6 AN5/DAN2- — Single-ended analog input/Differential analog input negative terminal

AN5 is a single-ended analog input to the two on-chip ADCs. DAN2- is the negative terminal of the differential analog input DAN2 (DAN2+ - DAN2-).

30.4.2.7 AN6/DAN3+ — Single-ended analog input/Differential analog input positive terminal

AN6 is a single-ended analog input to the two on-chip ADCs. DAN3+ is the positive terminal of the differential analog input DAN3 (DAN3+ - DAN3-).

30.4.2.8 AN7/DAN3- — Single-ended analog input/Differential analog input negative terminal

AN7 is a single-ended analog input to the two on-chip ADCs. DAN3- is the negative terminal of the differential analog input DAN3 (DAN3+ - DAN3-).

30.4.2.9 AN8/ANW — Single-ended analog input/ Single-ended analog input from external multiplexers

AN8 is a single-ended analog input to the two on-chip ADCs. ANW is a single-ended analog input to one of the on-chip ADCs in external multiplexed mode.

30.4.2.10 AN9/ANX/TBIAS — Single-ended analog input/ Single-ended analog input from external multiplexers/Test Bias

AN9 is a single-ended analog input to the two on-chip ADCs. ANX is a single-ended analog input to one of the on-chip ADCs in external multiplexed mode. TBIAS is used during factory test to verify the bias generator circuit.

30.4.2.11 AN10/ANY — Single-ended analog input/ Single-ended analog input from external multiplexers

AN10 is a single-ended analog input to the two on-chip ADCs. ANY is a single-ended analog input to one of the on-chip ADCs in external multiplexed mode.

30.4.2.12 AN11/ANZ — Single-ended analog input/ Single-ended analog input from external multiplexers

AN11 is a single-ended analog input to the two on-chip ADCs. ANZ is a single-ended analog input to one of the on-chip ADCs in external multiplexed mode.

30.4.2.13 AN12/T50PVREF — Single-ended analog input/ Test 50% VREF analog output

AN12 is a single-ended analog input to the two on-chip ADCs. T50PVREF is used during ADC factory test.

30.4.2.14 AN13/T25PVREF — Single-ended analog input/ Test 25% VREF analog output

AN13 is a single-ended analog input to the two on-chip ADCs. T25PVREF is used during ADC factory test.

30.4.2.15 AN14/T75PVREF — Single-ended analog input/ Test 75% VREF analog output

AN14 is a single-ended analog input to the two on-chip ADCs. T75PVREF is used during ADC factory test.

30.4.2.16 AN15 — Single-ended analog input

AN15 is a single-ended analog inputs to the two on-chip ADCs.

30.4.2.17 AN16/ANR — Single-ended analog input/ Single-ended analog input from external multiplexers

AN16 is a single-ended analog input to the two on-chip ADCs. ANR is a single-ended analog input to one of the on-chip ADCs in external multiplexed mode. The external multiplexing capability is a device option.

30.4.2.18 AN17/ANS — Single-ended analog input/ Single-ended analog input from external multiplexers

AN17 is a single-ended analog input to the two on-chip ADCs. ANS is a single-ended analog input to one of the on-chip ADCs in external multiplexed mode. The external multiplexing capability is a device option.

30.4.2.19 AN18/ANT — Single-ended analog input/ Single-ended analog input from external multiplexers

AN18 is a single-ended analog input to the two on-chip ADCs. ANT is a single-ended analog input to one of the on-chip ADCs in external multiplexed mode. The external multiplexing capability is a device option.

30.4.2.20 AN19/ANU — Single-ended analog input/ Single-ended analog input from external multiplexers

AN19 is a single-ended analog input to the two on-chip ADCs. ANU is a single-ended analog input to one of the on-chip ADCs in external multiplexed mode. The external multiplexing capability is a device option.

30.4.2.21 AN20-AN39 — Single-ended analog input

AN20 through AN39 are single-ended analog inputs to the two on-chip ADCs.

30.4.2.22 INA_ADC0_0 - INA_ADC0_9 — Single-ended analog input

INA_ADC0_0 through INA_ADC0_9 are single-ended analog inputs to the on-chip ADC0.

30.4.2.23 INA_ADC1_0 - INA_ADC1_9 — Single-ended analog input

INA_ADC1_0 through INA_ADC1_9 are single-ended analog inputs to the on-chip ADC1.

30.4.2.24 MA0-MA2 — External multiplexer control signals

MA0, MA1, and MA2 combined form a select signal associated with external multiplexers.

30.4.2.25 VRH, VRL — Voltage reference high and voltage reference low

VRH and VRL are voltage references for the ADCs. VRH is the highest voltage reference, while VRL is the lowest voltage reference.

30.4.2.26 VDDA, VSSA — 5V VDD and VSS for the 5V analog components

VDDA is the positive power supply pin for the ADCs and VSSA is the negative power supply pin for the ADCs. Refer to electrical specifications.

30.4.2.27 REFBYPC — Reference Bypass Capacitor

The REFBYPC pin is used to connect an external bypass capacitor between REFBYPC and VRL. See section "Chip-specific EQADC information" for the capacitor value. This bypass capacitor is used to provide a stable reference voltage for the ADC.

NOTE

Prior EQADC versions have had two of these pins. To achieve accurate calibration, more time may be required with only one pin.

30.4.2.28 ETRIG0-ETRIG5 — External triggers

The external trigger signals are for hardware triggering. The EQADC can detect rising edge, falling edge, high level and low level on each of the external trigger signals. ETRIGx triggers CFIFOx. The EQADC also supports configurable digital filters for these external trigger signals. These digital filters can be bypassed by individual input control signals called eqadc_intern_trig_selx.

30.5 Memory map and register definition

This section provides memory maps and detailed descriptions of all registers. Data written to or read from reserved areas of the memory map is undefined.

This section provides memory maps and detailed descriptions of all registers. Data written to or read from reserved areas of the memory map is undefined.

EQADC memory map

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/page
0	EQADC Module Configuration Register (EQADC_MCR)	32	R/W	0000_0000h	30.5.1/1106

Table continues on the next page...

EQADC memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
4	EQADC Test Register (EQADC_TST)	32	R	0000_0000h	30.5.2/1107
C	EQADC External Trigger Digital Filter Register (EQADC_ETDFR)	32	R/W	0000_0000h	30.5.3/1108
10	EQADC CFIFO Push Register (EQADC_CFPR0)	32	W	0000_0000h	30.5.4/1109
14	EQADC CFIFO Push Register (EQADC_CFPR1)	32	W	0000_0000h	30.5.4/1109
18	EQADC CFIFO Push Register (EQADC_CFPR2)	32	W	0000_0000h	30.5.4/1109
1C	EQADC CFIFO Push Register (EQADC_CFPR3)	32	W	0000_0000h	30.5.4/1109
20	EQADC CFIFO Push Register (EQADC_CFPR4)	32	W	0000_0000h	30.5.4/1109
24	EQADC CFIFO Push Register (EQADC_CFPR5)	32	W	0000_0000h	30.5.4/1109
30	EQADC Result FIFO Pop Register (EQADC_RFPR0)	32	R	0000_0000h	30.5.5/1110
34	EQADC Result FIFO Pop Register (EQADC_RFPR1)	32	R	0000_0000h	30.5.5/1110
38	EQADC Result FIFO Pop Register (EQADC_RFPR2)	32	R	0000_0000h	30.5.5/1110
3C	EQADC Result FIFO Pop Register (EQADC_RFPR3)	32	R	0000_0000h	30.5.5/1110
40	EQADC Result FIFO Pop Register (EQADC_RFPR4)	32	R	0000_0000h	30.5.5/1110
44	EQADC Result FIFO Pop Register (EQADC_RFPR5)	32	R	0000_0000h	30.5.5/1110
50	EQADC CFIFO Control Register 0 (EQADC_CFCR0)	32	R/W	0000_0000h	30.5.6/1111
54	EQADC CFIFO Control Register 1 (EQADC_CFCR1)	32	R/W	0000_0000h	30.5.7/1115
58	EQADC CFIFO Control Register 2 (EQADC_CFCR2)	32	R/W	0000_0000h	30.5.8/1118
60	EQADC Interrupt and DMA Control Register 0 (EQADC_IDCR0)	32	R/W	0000_0000h	30.5.9/1121
64	EQADC Interrupt and DMA Control Register 1 (EQADC_IDCR1)	32	R/W	0000_0000h	30.5.10/1125
68	EQADC Interrupt and DMA Control Register 2 (EQADC_IDCR2)	32	R/W	0000_0000h	30.5.11/1130
70	EQADC FIFO and Interrupt Status Register (EQADC_FISR0)	32	R/W	0200_0000h	30.5.12/1135
74	EQADC FIFO and Interrupt Status Register (EQADC_FISR1)	32	R/W	0200_0000h	30.5.12/1135
78	EQADC FIFO and Interrupt Status Register (EQADC_FISR2)	32	R/W	0200_0000h	30.5.12/1135
7C	EQADC FIFO and Interrupt Status Register (EQADC_FISR3)	32	R/W	0200_0000h	30.5.12/1135
80	EQADC FIFO and Interrupt Status Register (EQADC_FISR4)	32	R/W	0200_0000h	30.5.12/1135
84	EQADC FIFO and Interrupt Status Register (EQADC_FISR5)	32	R/W	0200_0000h	30.5.12/1135
90	EQADC CFIFO Transfer Counter Register 0 (EQADC_CFTCR0)	32	R/W	0000_0000h	30.5.13/1139
94	EQADC CFIFO Transfer Counter Register 1 (EQADC_CFTCR1)	32	R/W	0000_0000h	30.5.14/1140
98	EQADC CFIFO Transfer Counter Register 2 (EQADC_CFTCR2)	32	R/W	0000_0000h	30.5.15/1141

Table continues on the next page...

EQADC memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
A0	EQADC CFIFO Status Snapshot Register 0 (EQADC_CFSSR0)	32	R	0000_7800h	30.5.16/1142
A4	EQADC CFIFO Status Snapshot Register 1 (EQADC_CFSSR1)	32	R	0000_7800h	30.5.17/1144
AC	EQADC CFIFO Status Register (EQADC_CFSR)	32	R	0000_0000h	30.5.18/1146
D0	EQADC STAC Bus Client Configuration Register (EQADC_STACCCR)	32	R/W	0000_0000h	30.5.19/1147
100	EQADC CFIFO0 Register (EQADC_CF0R0)	32	R	0000_0000h	30.5.20/1148
104	EQADC CFIFO0 Register (EQADC_CF0R1)	32	R	0000_0000h	30.5.20/1148
108	EQADC CFIFO0 Register (EQADC_CF0R2)	32	R	0000_0000h	30.5.20/1148
10C	EQADC CFIFO0 Register (EQADC_CF0R3)	32	R	0000_0000h	30.5.20/1148
110	EQADC CFIFO0 Extension Register (EQADC_CF0ER0)	32	R	0000_0000h	30.5.21/1149
114	EQADC CFIFO0 Extension Register (EQADC_CF0ER1)	32	R	0000_0000h	30.5.21/1149
118	EQADC CFIFO0 Extension Register (EQADC_CF0ER2)	32	R	0000_0000h	30.5.21/1149
11C	EQADC CFIFO0 Extension Register (EQADC_CF0ER3)	32	R	0000_0000h	30.5.21/1149
140	EQADC CFIFO1 Register (EQADC_CF1R0)	32	R	0000_0000h	30.5.22/1149
144	EQADC CFIFO1 Register (EQADC_CF1R1)	32	R	0000_0000h	30.5.22/1149
148	EQADC CFIFO1 Register (EQADC_CF1R2)	32	R	0000_0000h	30.5.22/1149
14C	EQADC CFIFO1 Register (EQADC_CF1R3)	32	R	0000_0000h	30.5.22/1149
180	EQADC CFIFO2 Register (EQADC_CF2R0)	32	R	0000_0000h	30.5.23/1150
184	EQADC CFIFO2 Register (EQADC_CF2R1)	32	R	0000_0000h	30.5.23/1150
188	EQADC CFIFO2 Register (EQADC_CF2R2)	32	R	0000_0000h	30.5.23/1150
18C	EQADC CFIFO2 Register (EQADC_CF2R3)	32	R	0000_0000h	30.5.23/1150
1C0	EQADC CFIFO3 Register (EQADC_CF3R0)	32	R	0000_0000h	30.5.24/1151
1C4	EQADC CFIFO3 Register (EQADC_CF3R1)	32	R	0000_0000h	30.5.24/1151

Table continues on the next page...

EQADC memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
1C8	EQADC CFIFO3 Register (EQADC_CF3R2)	32	R	0000_0000h	30.5.24/1151
1CC	EQADC CFIFO3 Register (EQADC_CF3R3)	32	R	0000_0000h	30.5.24/1151
200	EQADC CFIFO4 Register (EQADC_CF4R0)	32	R	0000_0000h	30.5.25/1151
204	EQADC CFIFO4 Register (EQADC_CF4R1)	32	R	0000_0000h	30.5.25/1151
208	EQADC CFIFO4 Register (EQADC_CF4R2)	32	R	0000_0000h	30.5.25/1151
20C	EQADC CFIFO4 Register (EQADC_CF4R3)	32	R	0000_0000h	30.5.25/1151
240	EQADC CFIFO5 Register (EQADC_CF5R0)	32	R	0000_0000h	30.5.26/1152
244	EQADC CFIFO5 Register (EQADC_CF5R1)	32	R	0000_0000h	30.5.26/1152
248	EQADC CFIFO5 Register (EQADC_CF5R2)	32	R	0000_0000h	30.5.26/1152
24C	EQADC CFIFO5 Register (EQADC_CF5R3)	32	R	0000_0000h	30.5.26/1152
300	EQADC RFIFO0 Register (EQADC_RF0R0)	32	R	0000_0000h	30.5.27/1153
304	EQADC RFIFO0 Register (EQADC_RF0R1)	32	R	0000_0000h	30.5.27/1153
308	EQADC RFIFO0 Register (EQADC_RF0R2)	32	R	0000_0000h	30.5.27/1153
30C	EQADC RFIFO0 Register (EQADC_RF0R3)	32	R	0000_0000h	30.5.27/1153
340	EQADC RFIFO1 Register (EQADC_RF1R0)	32	R	0000_0000h	30.5.28/1153
344	EQADC RFIFO1 Register (EQADC_RF1R1)	32	R	0000_0000h	30.5.28/1153
348	EQADC RFIFO1 Register (EQADC_RF1R2)	32	R	0000_0000h	30.5.28/1153
34C	EQADC RFIFO1 Register (EQADC_RF1R3)	32	R	0000_0000h	30.5.28/1153
380	EQADC RFIFO2 Register (EQADC_RF2R0)	32	R	0000_0000h	30.5.29/1154
384	EQADC RFIFO2 Register (EQADC_RF2R1)	32	R	0000_0000h	30.5.29/1154
388	EQADC RFIFO2 Register (EQADC_RF2R2)	32	R	0000_0000h	30.5.29/1154
38C	EQADC RFIFO2 Register (EQADC_RF2R3)	32	R	0000_0000h	30.5.29/1154

Table continues on the next page...

EQADC memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
3C0	EQADC RFIFO3 Register (EQADC_RF3R0)	32	R	0000_0000h	30.5.30/1155
3C4	EQADC RFIFO3 Register (EQADC_RF3R1)	32	R	0000_0000h	30.5.30/1155
3C8	EQADC RFIFO3 Register (EQADC_RF3R2)	32	R	0000_0000h	30.5.30/1155
3CC	EQADC RFIFO3 Register (EQADC_RF3R3)	32	R	0000_0000h	30.5.30/1155
400	EQADC RFIFO4 Register (EQADC_RF4R0)	32	R	0000_0000h	30.5.31/1155
404	EQADC RFIFO4 Register (EQADC_RF4R1)	32	R	0000_0000h	30.5.31/1155
408	EQADC RFIFO4 Register (EQADC_RF4R2)	32	R	0000_0000h	30.5.31/1155
40C	EQADC RFIFO4 Register (EQADC_RF4R3)	32	R	0000_0000h	30.5.31/1155
440	EQADC RFIFO5 Register (EQADC_RF5R0)	32	R	0000_0000h	30.5.32/1156
444	EQADC RFIFO5 Register (EQADC_RF5R1)	32	R	0000_0000h	30.5.32/1156
448	EQADC RFIFO5 Register (EQADC_RF5R2)	32	R	0000_0000h	30.5.32/1156
44C	EQADC RFIFO5 Register (EQADC_RF5R3)	32	R	0000_0000h	30.5.32/1156

30.5.1 EQADC Module Configuration Register (EQADC_MCR)

The EQADC Module Configuration Register (EQADC_MCR) contains bits used to control how the EQADC responds to a debug mode entry request.

Address: 0h base + 0h offset = 0h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0								ICEA0	ICEA1	DAM	0		0	DBG	
W									ICEA0	ICEA1	DAM					
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

EQADC_MCR field descriptions

Field	Description
0–23 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
24 ICEA0	Immediate Conversion Command Enable ADC0 ICEA0 enables the EQADC to abort on-chip ADC0 current conversion and to start the immediate conversion command from CFIFO0 in the requested ADC0. 0 Disable immediate conversion command request. 1 Enable immediate conversion command request.
25 ICEA1	Immediate Conversion Command Enable ADC1 ICEA1 enables the EQADC to abort on-chip ADC1 current conversion and to start the immediate conversion command from CFIFO0 in the requested ADC1. 0 Disable immediate conversion command request. 1 Enable immediate conversion command request.
26 DAM	Alternate Configuration Destination Addressing Mode DAM selects between two modes of sending the same conversion result to two different destinations. This is applied mainly when using Alternate Conversion Commands.

Table continues on the next page...

EQADC_MCR field descriptions (continued)

Field	Description
0	Legacy mode. Uses a defined set of pairs of companion module destinations in the PSI as defined by Table 30-121 in Alternate Configuration 1-14 Control Registers (ADC_ACR1-14) . The Extended Alternate Configuration registers (ADC_EACR1-14) are not considered.
1	Selects the Extended Alternate Configuration Registers to send result to two destinations. These destinations can be RFIFOs and companion modules connected to the PSI. See Extended Alternate Configuration 1-14 Control Registers (ADC_EACR1-14) for details.
27–28 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
29 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
30–31 DBG	Debug enable The DBG field defines the EQADC response to a debug mode entry request. . 0b00 Do not enter debug mode. 0b01 Reserved. 0b10 Enter debug mode. 0b11 Enter debug mode.

30.5.2 EQADC Test Register (EQADC_TST)

The EQADC Test Register (EQADC_TST) is used for test purposes only. This register can only be read/written in a test access, accessing the EQADC_TST register in any other way will result in a transfer error. In a non-test access to the EQADC_TST register, read data is undefined and written data is ignored.

Address: 0h base + 4h offset = 4h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	Reserved																															
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	

EQADC_TST field descriptions

Field	Description
0–31 Reserved	This field is reserved.

30.5.3 EQADC External Trigger Digital Filter Register (EQADC_ETDFR)

The EQADC External Trigger Digital Filter Register (EQADC_ETDFR) is used to set the minimum time a signal must be held in a logic state on the CFIFO triggers inputs to be recognized as an edge or level gated trigger. The Digital Filter Length field specifies the minimum number of system clocks that must be counted by the digital filter counter to recognize a logic state change. However, there is a control signal that can be used to bypass the digital filter when this is not needed.

The count specifies the sample period of the digital filter which is calculated according to the following equation:

FilterPeriod = (*SystemClockPeriod* x 2^{*DFL*}) + 1 (*SystemClockPeriod*)

Minimum clock counts for which an ETRIG signal needs to be stable to be passed through the filter are shown in [Table 30-9](#) . See [External trigger event detection](#) for more information on the digital filter.

NOTE

The DFL field must only be written when the MODEx of all CFIFOs are configured to disabled.

NOTE

When the digital filter is bypassed by using the input control, the DFL is not considered and the trigger input signal is not filtered.

Address: 0h base + Ch offset = Ch

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	
R	0																											DFL					
W																																	
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	

EQADC_ETDFR field descriptions

Field	Description
0–27 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
28–31 DFL	Digital Filter Length

Table continues on the next page...

EQADC_ETDFR field descriptions (continued)

Field	Description		
	The DFL field specifies the minimum number of system clocks that must be counted by the digital filter counter to recognize a logic state change.		
	DFL[0:3]	Minimum Clock Count	Minimum Time (ns) (system clock = 120 MHz)
	0b0000	2	16.66
	0b0001	3	25.00
	0b0010	5	41.66
	0b0011	9	75.00
	0b0100	17	141.66
	0b0101	33	275.00
	0b0110	65	541.66
	0b0111	129	1075.00
	0b1000	257	2141.66
	0b1001	513	4275.00
	0b1010	1025	8541.66
	0b1011	2049	17075.00
	0b1100	4097	34141.00
	0b1101	8193	68275.00
	0b1110	16385	136541.66
	0b1111	32769	273075.00

30.5.4 EQADC CFIFO Push Register (EQADC_CFPRn)

The EQADC CFIFO Push Registers (EQADC_CFPR) provide a mechanism to fill the CFIFOs with command messages from the CQueues. Refer to [EQADC command FIFOs](#) for more information on the CFIFOs and to [Message format in EQADC](#) for a description on command message formats.

NOTE

Only whole words must be written to EQADC_CFPR. Writing half-words or bytes to EQADC_CFPR will still push the whole 32-bit CF_PUSH field into the corresponding CFIFO, but undefined data will fill the areas of CF_PUSH that were not specifically designated as target locations for the write.

Memory map and register definition

Address: 0h base + 10h offset + (4d × i), where i=0d to 5d

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0																															
W	CF_PUSHX																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	

EQADC_CFPR_n field descriptions

Field	Description
0–31 CF_PUSHX	CFIFO Push Data x When CFIFO_x is not full, writing to the whole word or any bytes of EQADC_CFPR_x will push the 32-bit CF_PUSH_x value into CFIFO_x. Writing to the CF_PUSH_x field also increments the corresponding CFCTR_x value by one in EQADC FIFO and Interrupt Status Register (EQADC_FISR_n). When the CFIFO_x is full, the EQADC ignores any write to the CF_PUSH_x. Reading the EQADC_CFPR_x always returns zero.

30.5.5 EQADC Result FIFO Pop Register (EQADC_RFPR_n)

The EQADC Result FIFO Pop Registers (EQADC_RFPR) provide a mechanism to retrieve data from RFIFOs.

Address: 0h base + 30h offset + (4d × i), where i=0d to 5d

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0																RF_POPX															
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	

EQADC_RFPR_n field descriptions

Field	Description
0–15 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
16–31 RF_POPX	Result FIFO Pop Data x When RFIFO_x is not empty, the RF_POP_x contains the next unread entry value of RFIFO_x. Reading a word, a half-word, or any bytes from EQADC_RFPR_x will pop one entry from RFIFO_x and cause the RFCTR_x field to be decremented by one in EQADC FIFO and Interrupt Status Register (EQADC_FISR_n). When the RFIFO_x is empty, any read on EQADC_RFPR_x returns undefined data value and does not decrement the RFCTR_x value. Writing to EQADC_RFPR_x has no effect.

30.5.6 EQADC CFIFO Control Register 0 (EQADC_CFCR0)

The EQADC CFIFO Control Registers (EQADC_CFCR) contain bits that affect CFIFOs. These bits specify the CFIFO operation mode and can invalidate all of the CFIFO contents.

Address: 0h base + 50h offset = 50h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0			CFEEE0	STRME0	SSE0	CFINV0	0	MODE0				AMODE0			
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0						CFINV1	0	MODE1				0			
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

EQADC_CFCR0 field descriptions

Field	Description
0–2 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
3 CFEEE0	CFIFO0 Entry Number Extension Enable The CFEEE0 bit is used to enable the extension of the CFIFO0 entries. When in extended mode, the CFIFO0 total entries is the sum of normal entries plus the defined value for extension. For more details, see CFIFO0 streaming mode description . 0 CFIFO0 has a normal value of entries. 1 Enable the extension of CFIFO0 entries.
4 STRME0	CFIFO0 Streaming Mode Operation Enable The STRME0 bit is used to enable the streaming mode of operation of CFIFO0. In this case, it is possible to repeat some sequence of commands of this FIFO. For more details, refer to CFIFO0 streaming mode description . 0 Streaming mode of CFIFO0 is disabled. 1 Enable the streaming mode of CFIFO0.
5 SSE0	CFIFO Single-Scan Enable Bit 0

Table continues on the next page...

EQADC_CFCR0 field descriptions (continued)

Field	Description
	The SSEx bit is used to set the SSSx bit in EQADC FIFO and Interrupt Status Register (EQADC_FISRn) . Writing a "1" to SSEx will set the SSSx in EQADC FIFO and Interrupt Status Register (EQADC_FISRn), if the CFIFO is in single-scan mode. When SSSx is already asserted, writing a "1" to SSEx also has no effect. If the CFIFO is in continuous-scan mode or is disabled, writing a "1" to SSEx will not set SSSx. Writing a "0" to SSEx has no effect. SSEx always is read as "0". 0 No effect. 1 Set the SSSx bit.
6 CFINV0	CFIFO Invalidate Bit 0 The CFINVx bit causes the EQADC to invalidate all entries of CFIFOx. Writing a "1" to CFINVx will reset the value of CFCTRx in EQADC FIFO and Interrupt Status Register (EQADC_FISRn) . Writing a "1" to CFINVx also resets the Push Next Data Pointer, Transfer Next Data Pointer to the first entry of CFIFOx in Figure 30-123. CFINVx always is read as "0". Writing a "0" has no effect. NOTE: Writing CFINVx only invalidates commands stored in CFIFOx; previously transferred commands that are waiting for execution, that is commands stored in the CBuffers, will still be executed, and results generated by them will be stored in the appropriate RFIFO. NOTE: CFINVx must not be written unless the MODEx is configured to disabled, and CFIFO status is IDLE. 0 No effect. 1 Invalidate all of the entries in the corresponding CFIFO.
7 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
8–11 MODE0	CFIFO Operation Mode 0 The MODEx field selects the CFIFO operation mode for CFIFOx. See CFIFO scan trigger modes for more information on CFIFO trigger mode. NOTE: If MODEx is not disabled, it must not be changed to any other mode besides disabled. If MODEx is disabled and the CFIFO status is IDLE, MODEx can be changed to any other mode. 0000 Disabled 0001 Software Trigger, Single Scan 0010 Low Level Gated External Trigger, Single Scan 0011 High Level Gated External Trigger, Single Scan 0100 Falling Edge External Trigger, Single Scan 0101 Rising Edge External Trigger, Single Scan 0110 Falling or Rising Edge External Trigger, Single Scan 0111 Reserved 1000 Reserved 1001 Software Trigger, Continuous Scan 1010 Low Level Gated External Trigger, Continuous Scan 1011 High Level Gated External Trigger, Continuous Scan 1100 Falling Edge External Trigger, Continuous Scan 1101 Rising Edge External Trigger, Continuous Scan 1110 Falling or Rising Edge External Trigger, Continuous Scan 1111 Reserved
12–15 AMODE0	CFIFO0 Advance Trigger Operation Mode 0

Table continues on the next page...

EQADC_CFCR0 field descriptions (continued)

Field	Description
	The AMODE0 field selects the trigger mode for the ATRIG0 trigger signal in streaming mode. The use of reserved values drives to unknown behavior of the block. NOTE: If AMODE0 is not disabled, it must not be changed to any other mode besides disabled. If AMODE0 is disabled and the CFIFO0 status is IDLE, AMODE0 can be changed to any other mode. NOTE: For the streaming mode of operation when the ATRIG0 is used to enable the ETRIG0 or to advance the command queue, the normal mode of operation is external trigger single scan. Other settings are not fully tested. 0000 Disabled 0001 Software Trigger, Single Scan 0010 Low Level Gated External Trigger, Single Scan 0011 High Level Gated External Trigger, Single Scan 0100 Falling Edge External Trigger, Single Scan 0101 Rising Edge External Trigger, Single Scan 0110 Falling or Rising Edge External Trigger, Single Scan 0111 Reserved 1000 Reserved 1001 Reserved 1010 Reserved 1011 Reserved 1100 Reserved 1101 Reserved 1110 Reserved 1111 Reserved
16–20 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
21 SSE1	CFIFO Single-Scan Enable Bit 1 The SSEx bit is used to set the SSSx bit in EQADC FIFO and Interrupt Status Register (EQADC_FISRn). Writing a "1" to SSEx will set the SSSx in EQADC FIFO and Interrupt Status Register (EQADC_FISRn), if the CFIFO is in single-scan mode. When SSSx is already asserted, writing a "1" to SSEx also has no effect. If the CFIFO is in continuous-scan mode or is disabled, writing a "1" to SSEx will not set SSSx. Writing a "0" to SSEx has no effect. SSEx always is read as "0". 0 No effect. 1 Set the SSSx bit.
22 CFINV1	CFIFO Invalidate Bit 1 The CFINVx bit causes the EQADC to invalidate all entries of CFIFOx. Writing a "1" to CFINVx will reset the value of CFCTRx in EQADC FIFO and Interrupt Status Register (EQADC_FISRn). Writing a "1" to CFINVx also resets the Push Next Data Pointer, Transfer Next Data Pointer to the first entry of CFIFOx in Figure 30-123. CFINVx always is read as "0". Writing a "0" has no effect. NOTE: Writing CFINVx only invalidates commands stored in CFIFOx; previously transferred commands that are waiting for execution, that is commands stored in the CBuffers, will still be executed, and results generated by them will be stored in the appropriate RFIFO. NOTE: CFINVx must not be written unless the MODEx is configured to disabled, and CFIFO status is IDLE.

Table continues on the next page...

EQADC_CFCR0 field descriptions (continued)

Field	Description
	0 No effect. 1 Invalidate all of the entries in the corresponding CFIFO.
23 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
24–27 MODE1	CFIFO Operation Mode 1 The MODEx field selects the CFIFO operation mode for CFIFOx. See CFIFO scan trigger modes for more information on CFIFO trigger mode. NOTE: If MODEx is not disabled, it must not be changed to any other mode besides disabled. If MODEx is disabled and the CFIFO status is IDLE, MODEx can be changed to any other mode. 0000 Disabled 0001 Software Trigger, Single Scan 0010 Low Level Gated External Trigger, Single Scan 0011 High Level Gated External Trigger, Single Scan 0100 Falling Edge External Trigger, Single Scan 0101 Rising Edge External Trigger, Single Scan 0110 Falling or Rising Edge External Trigger, Single Scan 0111 Reserved 1000 Reserved 1001 Software Trigger, Continuous Scan 1010 Low Level Gated External Trigger, Continuous Scan 1011 High Level Gated External Trigger, Continuous Scan 1100 Falling Edge External Trigger, Continuous Scan 1101 Rising Edge External Trigger, Continuous Scan 1110 Falling or Rising Edge External Trigger, Continuous Scan 1111 Reserved
28–31 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

30.5.7 EQADC CFIFO Control Register 1 (EQADC_CFCR1)

The EQADC CFIFO Control Registers (EQADC_CFCR) contain bits that affect CFIFOs. These bits specify the CFIFO operation mode and can invalidate all of the CFIFO contents.

Address: 0h base + 54h offset = 54h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0							0	MODE2				0			
W						SSE2	CFINV2									
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0							0	MODE3				0			
W						SSE3	CFINV3									
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

EQADC_CFCR1 field descriptions

Field	Description
0–4 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
5 SSE2	CFIFO Single-Scan Enable Bit 2 The SSE _x bit is used to set the SSS _x bit in EQADC FIFO and Interrupt Status Register (EQADC_FISR_n) . Writing a "1" to SSE _x will set the SSS _x in EQADC FIFO and Interrupt Status Register (EQADC_FISR_n) , if the CFIFO is in single-scan mode. When SSS _x is already asserted, writing a "1" to SSE _x also has no effect. If the CFIFO is in continuous-scan mode or is disabled, writing a "1" to SSE _x will not set SSS _x . Writing a "0" to SSE _x has no effect. SSE _x always is read as "0". 0 No effect. 1 Set the SSS _x bit.
6 CFINV2	CFIFO Invalidate Bit 2 The CFINV _x bit causes the EQADC to invalidate all entries of CFIFO _x . Writing a "1" to CFINV _x will reset the value of CFCTR _x in EQADC FIFO and Interrupt Status Register (EQADC_FISR_n) . Writing a "1" to CFINV _x also resets the Push Next Data Pointer, Transfer Next Data Pointer to the first entry of CFIFO _x in Figure 30-123 . CFINV _x always is read as "0". Writing a "0" has no effect. NOTE: Writing CFINV _x only invalidates commands stored in CFIFO _x ; previously transferred commands that are waiting for execution, that is commands stored in the CBuffers, will still be executed, and results generated by them will be stored in the appropriate RFIFO. NOTE: CFINV _x must not be written unless the MODE _x is configured to disabled, and CFIFO status is IDLE.

Table continues on the next page...

EQADC_CFCR1 field descriptions (continued)

Field	Description
	0 No effect. 1 Invalidate all of the entries in the corresponding CFIFO.
7 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
8–11 MODE2	CFIFO Operation Mode 2 The MODEx field selects the CFIFO operation mode for CFIFOx. See CFIFO scan trigger modes for more information on CFIFO trigger mode. NOTE: If MODEx is not disabled, it must not be changed to any other mode besides disabled. If MODEx is disabled and the CFIFO status is IDLE, MODEx can be changed to any other mode. 0000 Disabled 0001 Software Trigger, Single Scan 0010 Low Level Gated External Trigger, Single Scan 0011 High Level Gated External Trigger, Single Scan 0100 Falling Edge External Trigger, Single Scan 0101 Rising Edge External Trigger, Single Scan 0110 Falling or Rising Edge External Trigger, Single Scan 0111 Reserved 1000 Reserved 1001 Software Trigger, Continuous Scan 1010 Low Level Gated External Trigger, Continuous Scan 1011 High Level Gated External Trigger, Continuous Scan 1100 Falling Edge External Trigger, Continuous Scan 1101 Rising Edge External Trigger, Continuous Scan 1110 Falling or Rising Edge External Trigger, Continuous Scan 1111 Reserved
12–20 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
21 SSE3	CFIFO Single-Scan Enable Bit 3 The SSEx bit is used to set the SSSx bit in EQADC FIFO and Interrupt Status Register (EQADC_FISRn) . Writing a "1" to SSEx will set the SSSx in EQADC FIFO and Interrupt Status Register (EQADC_FISRn) , if the CFIFO is in single-scan mode. When SSSx is already asserted, writing a "1" to SSEx also has no effect. If the CFIFO is in continuous-scan mode or is disabled, writing a "1" to SSEx will not set SSSx. Writing a "0" to SSEx has no effect. SSEx always is read as "0". 0 No effect. 1 Set the SSSx bit.
22 CFINV3	CFIFO Invalidate Bit 3 The CFINVx bit causes the EQADC to invalidate all entries of CFIFOx. Writing a "1" to CFINVx will reset the value of CFCTRx in EQADC FIFO and Interrupt Status Register (EQADC_FISRn) . Writing a "1" to CFINVx also resets the Push Next Data Pointer, Transfer Next Data Pointer to the first entry of CFIFOx in Figure 30-123 . CFINVx always is read as "0". Writing a "0" has no effect. NOTE: Writing CFINVx only invalidates commands stored in CFIFOx; previously transferred commands that are waiting for execution, that is commands stored in the CBuffers, will still be executed, and results generated by them will be stored in the appropriate RFIFO. NOTE: CFINVx must not be written unless the MODEx is configured to disabled, and CFIFO status is IDLE.

Table continues on the next page...

EQADC_CFCR1 field descriptions (continued)

Field	Description
	0 No effect. 1 Invalidate all of the entries in the corresponding CFIFO.
23 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
24–27 MODE3	CFIFO Operation Mode 3 The MODEx field selects the CFIFO operation mode for CFIFOx. See CFIFO scan trigger modes for more information on CFIFO trigger mode. NOTE: If MODEx is not disabled, it must not be changed to any other mode besides disabled. If MODEx is disabled and the CFIFO status is IDLE, MODEx can be changed to any other mode. 0000 Disabled 0001 Software Trigger, Single Scan 0010 Low Level Gated External Trigger, Single Scan 0011 High Level Gated External Trigger, Single Scan 0100 Falling Edge External Trigger, Single Scan 0101 Rising Edge External Trigger, Single Scan 0110 Falling or Rising Edge External Trigger, Single Scan 0111 Reserved 1000 Reserved 1001 Software Trigger, Continuous Scan 1010 Low Level Gated External Trigger, Continuous Scan 1011 High Level Gated External Trigger, Continuous Scan 1100 Falling Edge External Trigger, Continuous Scan 1101 Rising Edge External Trigger, Continuous Scan 1110 Falling or Rising Edge External Trigger, Continuous Scan 1111 Reserved
28–31 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

30.5.8 EQADC CFIFO Control Register 2 (EQADC_CFCR2)

The EQADC CFIFO Control Registers (EQADC_CFCR) contain bits that affect CFIFOs. These bits specify the CFIFO operation mode and can invalidate all of the CFIFO contents.

Address: 0h base + 58h offset = 58h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0							0	MODE4				0			
W						SSE4	CFINV4									
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0							0	MODE5				0			
W						SSE5	CFINV5									
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

EQADC_CFCR2 field descriptions

Field	Description
0–4 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
5 SSE4	CFIFO Single-Scan Enable Bit 4 The SSE _x bit is used to set the SSS _x bit in EQADC FIFO and Interrupt Status Register (EQADC_FISR_n) . Writing a "1" to SSE _x will set the SSS _x in EQADC FIFO and Interrupt Status Register (EQADC_FISR_n) , if the CFIFO is in single-scan mode. When SSS _x is already asserted, writing a "1" to SSE _x also has no effect. If the CFIFO is in continuous-scan mode or is disabled, writing a "1" to SSE _x will not set SSS _x . Writing a "0" to SSE _x has no effect. SSE _x always is read as "0". 0 No effect. 1 Set the SSS _x bit.
6 CFINV4	CFIFO Invalidate Bit 4 The CFINV _x bit causes the EQADC to invalidate all entries of CFIFO _x . Writing a "1" to CFINV _x will reset the value of CFCTR _x in EQADC FIFO and Interrupt Status Register (EQADC_FISR_n) . Writing a "1" to CFINV _x also resets the Push Next Data Pointer, Transfer Next Data Pointer to the first entry of CFIFO _x in Figure 30-123 . CFINV _x always is read as "0". Writing a "0" has no effect. NOTE: Writing CFINV _x only invalidates commands stored in CFIFO _x ; previously transferred commands that are waiting for execution, that is commands stored in the CBuffers, will still be executed, and results generated by them will be stored in the appropriate RFIFO. NOTE: CFINV _x must not be written unless the MODE _x is configured to disabled, and CFIFO status is IDLE.

Table continues on the next page...

EQADC_CFCR2 field descriptions (continued)

Field	Description
	0 No effect. 1 Invalidate all of the entries in the corresponding CFIFO.
7 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
8–11 MODE4	CFIFO Operation Mode 4 The MODEx field selects the CFIFO operation mode for CFIFOx. See CFIFO scan trigger modes for more information on CFIFO trigger mode. NOTE: If MODEx is not disabled, it must not be changed to any other mode besides disabled. If MODEx is disabled and the CFIFO status is IDLE, MODEx can be changed to any other mode. 0000 Disabled 0001 Software Trigger, Single Scan 0010 Low Level Gated External Trigger, Single Scan 0011 High Level Gated External Trigger, Single Scan 0100 Falling Edge External Trigger, Single Scan 0101 Rising Edge External Trigger, Single Scan 0110 Falling or Rising Edge External Trigger, Single Scan 0111 Reserved 1000 Reserved 1001 Software Trigger, Continuous Scan 1010 Low Level Gated External Trigger, Continuous Scan 1011 High Level Gated External Trigger, Continuous Scan 1100 Falling Edge External Trigger, Continuous Scan 1101 Rising Edge External Trigger, Continuous Scan 1110 Falling or Rising Edge External Trigger, Continuous Scan 1111 Reserved
12–20 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
21 SSE5	CFIFO Single-Scan Enable Bit 5 The SSEx bit is used to set the SSSx bit in EQADC FIFO and Interrupt Status Register (EQADC_FISRn) . Writing a "1" to SSEx will set the SSSx in EQADC FIFO and Interrupt Status Register (EQADC_FISRn) , if the CFIFO is in single-scan mode. When SSSx is already asserted, writing a "1" to SSEx also has no effect. If the CFIFO is in continuous-scan mode or is disabled, writing a "1" to SSEx will not set SSSx. Writing a "0" to SSEx has no effect. SSEx always is read as "0". 0 No effect. 1 Set the SSSx bit.
22 CFINV5	CFIFO Invalidate Bit 5 The CFINVx bit causes the EQADC to invalidate all entries of CFIFOx. Writing a "1" to CFINVx will reset the value of CFCTRx in EQADC FIFO and Interrupt Status Register (EQADC_FISRn) . Writing a "1" to CFINVx also resets the Push Next Data Pointer, Transfer Next Data Pointer to the first entry of CFIFOx in Figure 30-123 . CFINVx always is read as "0". Writing a "0" has no effect. NOTE: Writing CFINVx only invalidates commands stored in CFIFOx; previously transferred commands that are waiting for execution, that is commands stored in the CBuffers, will still be executed, and results generated by them will be stored in the appropriate RFIFO. NOTE: CFINVx must not be written unless the MODEx is configured to disabled, and CFIFO status is IDLE.

Table continues on the next page...

EQADC_CFCR2 field descriptions (continued)

Field	Description
	0 No effect. 1 Invalidate all of the entries in the corresponding CFIFO.
23 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
24–27 MODE5	CFIFO Operation Mode 5 The MODEx field selects the CFIFO operation mode for CFIFOx. See CFIFO scan trigger modes for more information on CFIFO trigger mode. NOTE: If MODEx is not disabled, it must not be changed to any other mode besides disabled. If MODEx is disabled and the CFIFO status is IDLE, MODEx can be changed to any other mode. 0000 Disabled 0001 Software Trigger, Single Scan 0010 Low Level Gated External Trigger, Single Scan 0011 High Level Gated External Trigger, Single Scan 0100 Falling Edge External Trigger, Single Scan 0101 Rising Edge External Trigger, Single Scan 0110 Falling or Rising Edge External Trigger, Single Scan 0111 Reserved 1000 Reserved 1001 Software Trigger, Continuous Scan 1010 Low Level Gated External Trigger, Continuous Scan 1011 High Level Gated External Trigger, Continuous Scan 1100 Falling Edge External Trigger, Continuous Scan 1101 Rising Edge External Trigger, Continuous Scan 1110 Falling or Rising Edge External Trigger, Continuous Scan 1111 Reserved
28–31 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

30.5.9 EQADC Interrupt and DMA Control Register 0 (EQADC_IDCR0)

The EQADC Interrupt Control Registers (EQADC_IDCR) contain bits to enable the generation of interrupt or DMA requests when the corresponding flag bits are set in [EQADC FIFO and Interrupt Status Register \(EQADC_FISR_n\)](#).

Address: 0h base + 60h offset = 60h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	NCIE0	TORIE0	PIE0	EOQIE0	CFUIE0	0	CFFE0	CFFS0	0				RFOIE0	0	RFDE0	RFDS0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	NCIE1	TORIE1	PIE1	EOQIE1	CFUIE1	0	CFFE1	CFFS1	0				RFOIE1	0	RFDE1	RFDS1
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

EQADC_IDCR0 field descriptions

Field	Description
0 NCIE0	Non-Coherency Interrupt Enable 0 NCIE_x enables the EQADC to generate an interrupt request when the corresponding NCF_x in EQADC FIFO and Interrupt Status Register (EQADC_FISR_n) is asserted. 0 Disable non-coherency interrupt request. 1 Enable non-coherency interrupt request.
1 TORIE0	Trigger Overrun Interrupt Enable 0 TORIE_x enables the EQADC to generate an interrupt request when the corresponding TORF_x in EQADC DMA/interrupt request is asserted. Apart from generating an independent interrupt request for a CFIFO_x Trigger Overrun event, the EQADC also provides a combined interrupt at which the Result FIFO Overflow Interrupt, the Command FIFO Underflow Interrupt, and the Command FIFO Trigger Overrun Interrupt requests of ALL CFIFOs are ORed. When RFOIE_x, CFUIE_x, and TORIE_x are all asserted, this combined interrupt request is asserted whenever one of the following 18 flags becomes asserted: RFOF_x, CFUF_x, and TORF_x (assuming that all interrupts are enabled). See EQADC DMA/interrupt request for details. 0 Disable trigger overrun interrupt request. 1 Enable trigger overrun interrupt request.
2 PIE0	Pause Interrupt Enable 0 PIE_x enables the EQADC to generate an interrupt request when the corresponding PF_x in EQADC FIFO and Interrupt Status Register (EQADC_FISR_n) is asserted.

Table continues on the next page...

EQADC_IDCR0 field descriptions (continued)

Field	Description
	0 Disable pause interrupt request. 1 Enable pause interrupt request.
3 EOQIE0	End of Queue Interrupt Enable 0 EOQIE0 enables the EQADC to generate an interrupt request when the corresponding EOQFx in EQADC FIFO and Interrupt Status Register (EQADC_FISRn) is asserted. 0 Disable End of Queue interrupt request. 1 Enable End of Queue interrupt request.
4 CFUIE0	CFIFO Underflow Interrupt Enable 0 CFUIE0 enables the EQADC to generate an interrupt request when the corresponding CFUFx in EQADC FIFO and Interrupt Status Register (EQADC_FISRn) is asserted. Apart from generating an independent interrupt request for a CFIFOx underflow event, the EQADC also provides a combined interrupt at which the Result FIFO Overflow Interrupt, the Command FIFO Underflow Interrupt, and the Command FIFO Trigger Overrun Interrupt requests of ALL CFIFOs are ORed. When RFOIE0, CFUIE0, and TORIE0 are all asserted, this combined interrupt request is asserted whenever one of the following 18 flags becomes asserted: RFOFx, CFUFx, and TORFx (assuming that all interrupts are enabled). See EQADC DMA/interrupt request for details. 0 Disable Underflow Interrupt request. 1 Enable Underflow Interrupt request.
5 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
6 CFFE0	CFIFO Fill Enable 0 CFFE0 enables the EQADC to generate an interrupt request (CFFSx is negated) or DMA request (CFFSx is asserted) when CFFFx in EQADC FIFO and Interrupt Status Register (EQADC_FISRn) is asserted. NOTE: CFFE0 must not be negated while a DMA transaction is in progress. 0 Disable CFIFO Fill DMA or Interrupt request. 1 Enable CFIFO Fill DMA or Interrupt request.
7 CFFS0	CFIFO Fill Select 0 CFFS0 selects if a DMA or interrupt request is generated when CFFFx in EQADC FIFO and Interrupt Status Register (EQADC_FISRn) is asserted. If CFFE0 is asserted, the EQADC generates an interrupt request when CFFS0 is negated, or it generates a DMA request if CFFS0 is asserted. NOTE: CFFS0 must not be negated while a DMA transaction is in progress. 0 Generate interrupt request to move data from the system memory to CFIFOx. 1 Generate DMA request to move data from the system memory to CFIFOx.
8–11 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
12 RFOIE0	RFIFO Overflow Interrupt Enable 0 RFOIE0 enables the EQADC to generate an interrupt request when the corresponding RFOFx in EQADC FIFO and Interrupt Status Register (EQADC_FISRn) is asserted. Apart from generating an independent interrupt request for an RFIFOx overflow event, the EQADC also provides a combined interrupt at which the Result FIFO Overflow Interrupt, the Command FIFO Underflow Interrupt, and the Command FIFO Trigger Overrun Interrupt requests of ALL CFIFOs are ORed. When

Table continues on the next page...

EQADC_IDCR0 field descriptions (continued)

Field	Description
	RFOIEx, CFUIEx, and TORIEx are all asserted, this combined interrupt request is asserted whenever one of the following 18 flags becomes asserted: RFOFx, CFUFx, and TORFx (assuming that all interrupts are enabled). See EQADC DMA/interrupt request for details. 0 Disable Overflow Interrupt request. 1 Enable Overflow Interrupt request.
13 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
14 RFDE0	RFIFO Drain Enable 0 RFDEx enables the EQADC to generate an interrupt request (RFDSx is negated) or DMA request (RFDSx is asserted) when RFDFx in EQADC FIFO and Interrupt Status Register (EQADC_FISRn) is asserted. NOTE: RFDEx must not be negated while a DMA transaction is in progress. 0 Disable RFIFO Drain DMA or Interrupt request. 1 Enable RFIFO Drain DMA or Interrupt request.
15 RFDS0	RFIFO Drain Select 0 RFDSx selects if a DMA or interrupt request is generated when RFDFx in EQADC FIFO and Interrupt Status Register (EQADC_FISRn) is asserted. If RFDEx is asserted, the EQADC generates an interrupt request when RFDSx is negated, or it generates a DMA request when RFDSx is asserted. NOTE: RFDSx must not be negated while a DMA transaction is in progress. 0 Generate interrupt request to move data from RFIFOx to the system memory. 1 Generate DMA request to move data from RFIFOx to the system memory.
16 NCIE1	Non-Coherency Interrupt Enable 1 NCIEx enables the EQADC to generate an interrupt request when the corresponding NCFx in EQADC FIFO and Interrupt Status Register (EQADC_FISRn) is asserted. 0 Disable non-coherency interrupt request. 1 Enable non-coherency interrupt request.
17 TORIE1	Trigger Overrun Interrupt Enable 1 TORIEx enables the EQADC to generate an interrupt request when the corresponding TORFx in EQADC FIFO and Interrupt Status Register (EQADC_FISRn) is asserted. Apart from generating an independent interrupt request for a CFIFOx Trigger Overrun event, the EQADC also provides a combined interrupt at which the Result FIFO Overflow Interrupt, the Command FIFO Underflow Interrupt, and the Command FIFO Trigger Overrun Interrupt requests of ALL CFIFOs are ORed. When RFOIEx, CFUIEx, and TORIEx are all asserted, this combined interrupt request is asserted whenever one of the following 18 flags becomes asserted: RFOFx, CFUFx, and TORFx (assuming that all interrupts are enabled). See EQADC DMA/interrupt request for details. 0 Disable trigger overrun interrupt request. 1 Enable trigger overrun interrupt request.
18 PIE1	Pause Interrupt Enable 1 PIEx enables the EQADC to generate an interrupt request when the corresponding PFx in EQADC FIFO and Interrupt Status Register (EQADC_FISRn) is asserted. 0 Disable pause interrupt request. 1 Enable pause interrupt request.

Table continues on the next page...

EQADC_IDCR0 field descriptions (continued)

Field	Description
19 EOQIE1	End of Queue Interrupt Enable 1 EOQIE1 enables the EQADC to generate an interrupt request when the corresponding EOQF1 in EQADC FIFO and Interrupt Status Register (EQADC_FISRn) is asserted. 0 Disable End of Queue interrupt request. 1 Enable End of Queue interrupt request.
20 CFUIE1	CFIFO Underflow Interrupt Enable 1 CFUIE1 enables the EQADC to generate an interrupt request when the corresponding CFUF1 in EQADC FIFO and Interrupt Status Register (EQADC_FISRn) is asserted. Apart from generating an independent interrupt request for a CFIFOx underflow event, the EQADC also provides a combined interrupt at which the Result FIFO Overflow Interrupt, the Command FIFO Underflow Interrupt, and the Command FIFO Trigger Overrun Interrupt requests of ALL CFIFOs are ORed. When RFOIE1, CFUIE1, and TORIE1 are all asserted, this combined interrupt request is asserted whenever one of the following 18 flags becomes asserted: RFOF1, CFUF1, and TORF1 (assuming that all interrupts are enabled). See EQADC DMA/interrupt request for details. 0 Disable Underflow Interrupt request. 1 Enable Underflow Interrupt request.
21 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
22 CFFE1	CFIFO Fill Enable 1 CFFE1 enables the EQADC to generate an interrupt request (CFFS1 is negated) or DMA request (CFFS1 is asserted) when CFFF1 in EQADC FIFO and Interrupt Status Register (EQADC_FISRn) is asserted. NOTE: CFFE1 must not be negated while a DMA transaction is in progress. 0 Disable CFIFO Fill DMA or Interrupt request. 1 Enable CFIFO Fill DMA or Interrupt request.
23 CFFS1	CFIFO Fill Select 1 CFFS1 selects if a DMA or interrupt request is generated when CFFF1 in EQADC FIFO and Interrupt Status Register (EQADC_FISRn) is asserted. If CFFE1 is asserted, the EQADC generates an interrupt request when CFFS1 is negated, or it generates a DMA request if CFFS1 is asserted. NOTE: CFFS1 must not be negated while a DMA transaction is in progress. 0 Generate interrupt request to move data from the system memory to CFIFOx. 1 Generate DMA request to move data from the system memory to CFIFOx.
24–27 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
28 RFOIE1	RFIFO Overflow Interrupt Enable 1 RFOIE1 enables the EQADC to generate an interrupt request when the corresponding RFOF1 in EQADC FIFO and Interrupt Status Register (EQADC_FISRn) is asserted. Apart from generating an independent interrupt request for an RFIFOx overflow event, the EQADC also provides a combined interrupt at which the Result FIFO Overflow Interrupt, the Command FIFO Underflow Interrupt, and the Command FIFO Trigger Overrun Interrupt requests of ALL CFIFOs are ORed. When RFOIE1, CFUIE1, and TORIE1 are all asserted, this combined interrupt request is asserted whenever one of the following 18 flags becomes asserted: RFOF1, CFUF1, and TORF1 (assuming that all interrupts are enabled). See EQADC DMA/interrupt request for details.

Table continues on the next page...

EQADC_IDCR0 field descriptions (continued)

Field	Description
	0 Disable Overflow Interrupt request. 1 Enable Overflow Interrupt request.
29 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
30 RFDE1	RFIFO Drain Enable 1 RFDEx enables the EQADC to generate an interrupt request (RFDSx is negated) or DMA request (RFDSx is asserted) when RFDFx in EQADC FIFO and Interrupt Status Register (EQADC_FISR_n) is asserted. NOTE: RFDEx must not be negated while a DMA transaction is in progress. 0 Disable RFIFO Drain DMA or Interrupt request. 1 Enable RFIFO Drain DMA or Interrupt request.
31 RFDS1	RFIFO Drain Select 1 RFDSx selects if a DMA or interrupt request is generated when RFDFx in EQADC FIFO and Interrupt Status Register (EQADC_FISR_n) is asserted. If RFDEx is asserted, the EQADC generates an interrupt request when RFDSx is negated, or it generates a DMA request when RFDSx is asserted. NOTE: RFDSx must not be negated while a DMA transaction is in progress. 0 Generate interrupt request to move data from RFIFOx to the system memory. 1 Generate DMA request to move data from RFIFOx to the system memory.

30.5.10 EQADC Interrupt and DMA Control Register 1 (EQADC_IDCR1)

The EQADC Interrupt Control Registers (EQADC_IDCR) contain bits to enable the generation of interrupt or DMA requests when the corresponding flag bits are set in [EQADC FIFO and Interrupt Status Register \(EQADC_FISR_n\)](#).

Address: 0h base + 64h offset = 64h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	NCIE2	TORIE2	PIE2	EOQIE2	CFUIE2	0	CFFE2	CFFS2	0				RFOIE2	0	RFDE2	RFDS2
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	NCIE3	TORIE3	PIE3	EOQIE3	CFUIE3	0	CFFE3	CFFS3	0				RFOIE3	0	RFDE3	RFDS3
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

EQADC_IDCR1 field descriptions

Field	Description
0 NCIE2	Non-Coherency Interrupt Enable 2 NCIEEx enables the EQADC to generate an interrupt request when the corresponding NCFx in EQADC FIFO and Interrupt Status Register (EQADC_FISRn) is asserted. 0 Disable non-coherency interrupt request. 1 Enable non-coherency interrupt request.
1 TORIE2	Trigger Overrun Interrupt Enable 2 TORIEEx enables the EQADC to generate an interrupt request when the corresponding TORFx in EQADC FIFO and Interrupt Status Register (EQADC_FISRn) is asserted. Apart from generating an independent interrupt request for a CFIFOx Trigger Overrun event, the EQADC also provides a combined interrupt at which the Result FIFO Overflow Interrupt, the Command FIFO Underflow Interrupt, and the Command FIFO Trigger Overrun Interrupt requests of ALL CFIFOs are ORed. When RFOIEEx, CFUIEx, and TORIEEx are all asserted, this combined interrupt request is asserted whenever one of the following 18 flags becomes asserted: RFOFx, CFUFx, and TORFx (assuming that all interrupts are enabled). See EQADC DMA/interrupt request for details. 0 Disable trigger overrun interrupt request. 1 Enable trigger overrun interrupt request.
2 PIE2	Pause Interrupt Enable 2 PIEEx enables the EQADC to generate an interrupt request when the corresponding PFx in EQADC FIFO and Interrupt Status Register (EQADC_FISRn) is asserted. 0 Disable pause interrupt request. 1 Enable pause interrupt request.
3 EOQIE2	End of Queue Interrupt Enable 2 EOQIEEx enables the EQADC to generate an interrupt request when the corresponding EOQFx in EQADC FIFO and Interrupt Status Register (EQADC_FISRn) is asserted. 0 Disable End of Queue interrupt request. 1 Enable End of Queue interrupt request.
4 CFUIE2	CFIFO Underflow Interrupt Enable 2 CFUIEx enables the EQADC to generate an interrupt request when the corresponding CFUFx in EQADC FIFO and Interrupt Status Register (EQADC_FISRn) is asserted. Apart from generating an independent interrupt request for a CFIFOx underflow event, the EQADC also provides a combined interrupt at which the Result FIFO Overflow Interrupt, the Command FIFO Underflow Interrupt, and the Command FIFO Trigger Overrun Interrupt requests of ALL CFIFOs are ORed. When RFOIEEx, CFUIEx, and TORIEEx are all asserted, this combined interrupt request is asserted whenever one of the following 18 flags becomes asserted: RFOFx, CFUFx, and TORFx (assuming that all interrupts are enabled). See EQADC DMA/interrupt request for details. 0 Disable Underflow Interrupt request. 1 Enable Underflow Interrupt request.
5 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
6 CFFE2	CFIFO Fill Enable 2 CFFEx enables the EQADC to generate an interrupt request (CFFSx is negated) or DMA request (CFFSx is asserted) when CFFFx in EQADC FIFO and Interrupt Status Register (EQADC_FISRn) is asserted. NOTE: CFFEx must not be negated while a DMA transaction is in progress.

Table continues on the next page...

EQADC_IDCR1 field descriptions (continued)

Field	Description
	0 Disable CFIFO Fill DMA or Interrupt request. 1 Enable CFIFO Fill DMA or Interrupt request.
7 CFFS2	CFIFO Fill Select 2 CFFSx selects if a DMA or interrupt request is generated when CFFEx in EQADC FIFO and Interrupt Status Register (EQADC_FISRn) is asserted. If CFFEx is asserted, the EQADC generates an interrupt request when CFFSx is negated, or it generates a DMA request if CFFSx is asserted. NOTE: CFFSx must not be negated while a DMA transaction is in progress. 0 Generate interrupt request to move data from the system memory to CFIFOx. 1 Generate DMA request to move data from the system memory to CFIFOx.
8–11 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
12 RFOIE2	RFIFO Overflow Interrupt Enable 2 RFOIEx enables the EQADC to generate an interrupt request when the corresponding RFOFx in EQADC FIFO and Interrupt Status Register (EQADC_FISRn) is asserted. Apart from generating an independent interrupt request for an RFIFOx overflow event, the EQADC also provides a combined interrupt at which the Result FIFO Overflow Interrupt, the Command FIFO Underflow Interrupt, and the Command FIFO Trigger Overrun Interrupt requests of ALL CFIFOs are ORed. When RFOIEx, CFUIEx, and TORIEx are all asserted, this combined interrupt request is asserted whenever one of the following 18 flags becomes asserted: RFOFx, CFUFx, and TORFx (assuming that all interrupts are enabled). See EQADC DMA/interrupt request for details. 0 Disable Overflow Interrupt request. 1 Enable Overflow Interrupt request.
13 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
14 RFDE2	RFIFO Drain Enable 2 RFDEx enables the EQADC to generate an interrupt request (RFDSx is negated) or DMA request (RFDSx is asserted) when RFDFx in EQADC FIFO and Interrupt Status Register (EQADC_FISRn) is asserted. NOTE: RFDEx must not be negated while a DMA transaction is in progress. 0 Disable RFIFO Drain DMA or Interrupt request. 1 Enable RFIFO Drain DMA or Interrupt request.
15 RFDS2	RFIFO Drain Select 2 RFDSx selects if a DMA or interrupt request is generated when RFDFx in EQADC FIFO and Interrupt Status Register (EQADC_FISRn) is asserted. If RFDEx is asserted, the EQADC generates an interrupt request when RFDSx is negated, or it generates a DMA request when RFDSx is asserted. NOTE: RFDSx must not be negated while a DMA transaction is in progress. 0 Generate interrupt request to move data from RFIFOx to the system memory. 1 Generate DMA request to move data from RFIFOx to the system memory.
16 NCIE3	Non-Coherency Interrupt Enable 3 NCIEx enables the EQADC to generate an interrupt request when the corresponding NCFx in EQADC FIFO and Interrupt Status Register (EQADC_FISRn) is asserted. 0 Disable non-coherency interrupt request. 1 Enable non-coherency interrupt request.

Table continues on the next page...

EQADC_IDCR1 field descriptions (continued)

Field	Description
17 TORIE3	Trigger Overrun Interrupt Enable 3 TORIE_x enables the EQADC to generate an interrupt request when the corresponding TORF_x in EQADC FIFO and Interrupt Status Register (EQADC_FISR_n) is asserted. Apart from generating an independent interrupt request for a CFIFO_x Trigger Overrun event, the EQADC also provides a combined interrupt at which the Result FIFO Overflow Interrupt, the Command FIFO Underflow Interrupt, and the Command FIFO Trigger Overrun Interrupt requests of ALL CFIFOs are ORed. When RFOIE_x, CFUIE_x, and TORIE_x are all asserted, this combined interrupt request is asserted whenever one of the following 18 flags becomes asserted: RFOF_x, CFUF_x, and TORF_x (assuming that all interrupts are enabled). See EQADC DMA/interrupt request for details. 0 Disable trigger overrun interrupt request. 1 Enable trigger overrun interrupt request.
18 PIE3	Pause Interrupt Enable 3 PIE_x enables the EQADC to generate an interrupt request when the corresponding PF_x in EQADC FIFO and Interrupt Status Register (EQADC_FISR_n) is asserted. 0 Disable pause interrupt request. 1 Enable pause interrupt request.
19 EOQIE3	End of Queue Interrupt Enable 3 EOQIE_x enables the EQADC to generate an interrupt request when the corresponding EOQF_x in EQADC FIFO and Interrupt Status Register (EQADC_FISR_n) is asserted. 0 Disable End of Queue interrupt request. 1 Enable End of Queue interrupt request.
20 CFUIE3	CFIFO Underflow Interrupt Enable 3 CFUIE_x enables the EQADC to generate an interrupt request when the corresponding CFUF_x in EQADC FIFO and Interrupt Status Register (EQADC_FISR_n) is asserted. Apart from generating an independent interrupt request for a CFIFO_x underflow event, the EQADC also provides a combined interrupt at which the Result FIFO Overflow Interrupt, the Command FIFO Underflow Interrupt, and the Command FIFO Trigger Overrun Interrupt requests of ALL CFIFOs are ORed. When RFOIE_x, CFUIE_x, and TORIE_x are all asserted, this combined interrupt request is asserted whenever one of the following 18 flags becomes asserted: RFOF_x, CFUF_x, and TORF_x (assuming that all interrupts are enabled). See EQADC DMA/interrupt request for details. 0 Disable Underflow Interrupt request. 1 Enable Underflow Interrupt request.
21 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
22 CFFE3	CFIFO Fill Enable 3 CFFE_x enables the EQADC to generate an interrupt request (CFFS_x is negated) or DMA request (CFFS_x is asserted) when CFFF_x in EQADC FIFO and Interrupt Status Register (EQADC_FISR_n) is asserted. NOTE: CFFE_x must not be negated while a DMA transaction is in progress. 0 Disable CFIFO Fill DMA or Interrupt request. 1 Enable CFIFO Fill DMA or Interrupt request.
23 CFFS3	CFIFO Fill Select 3

Table continues on the next page...

EQADC_IDCR1 field descriptions (continued)

Field	Description
	CFFSx selects if a DMA or interrupt request is generated when CFFFx in EQADC FIFO and Interrupt Status Register (EQADC_FISRn) is asserted. If CFFEx is asserted, the EQADC generates an interrupt request when CFFSx is negated, or it generates a DMA request if CFFSx is asserted. NOTE: CFFSx must not be negated while a DMA transaction is in progress. 0 Generate interrupt request to move data from the system memory to CFIF0x. 1 Generate DMA request to move data from the system memory to CFIF0x.
24–27 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
28 RFOIE3	RFIFO Overflow Interrupt Enable 3 RFOIEx enables the EQADC to generate an interrupt request when the corresponding RFOFx in EQADC FIFO and Interrupt Status Register (EQADC_FISRn) is asserted. Apart from generating an independent interrupt request for an RFIF0x overflow event, the EQADC also provides a combined interrupt at which the Result FIFO Overflow Interrupt, the Command FIFO Underflow Interrupt, and the Command FIFO Trigger Overrun Interrupt requests of ALL CFIF0s are ORed. When RFOIEx, CFUIEx, and TORIEx are all asserted, this combined interrupt request is asserted whenever one of the following 18 flags becomes asserted: RFOFx, CFUFx, and TORFx (assuming that all interrupts are enabled). See EQADC DMA/interrupt request for details. 0 Disable Overflow Interrupt request. 1 Enable Overflow Interrupt request.
29 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
30 RFDE3	RFIFO Drain Enable 3 RFDEx enables the EQADC to generate an interrupt request (RFDSx is negated) or DMA request (RFDSx is asserted) when RFDFx in EQADC FIFO and Interrupt Status Register (EQADC_FISRn) is asserted. NOTE: RFDEx must not be negated while a DMA transaction is in progress. 0 Disable RFIFO Drain DMA or Interrupt request. 1 Enable RFIFO Drain DMA or Interrupt request.
31 RFDS3	RFIFO Drain Select 3 RFDSx selects if a DMA or interrupt request is generated when RFDFx in EQADC FIFO and Interrupt Status Register (EQADC_FISRn) is asserted. If RFDEx is asserted, the EQADC generates an interrupt request when RFDSx is negated, or it generates a DMA request when RFDSx is asserted. NOTE: RFDSx must not be negated while a DMA transaction is in progress. 0 Generate interrupt request to move data from RFIF0x to the system memory. 1 Generate DMA request to move data from RFIF0x to the system memory.

30.5.11 EQADC Interrupt and DMA Control Register 2 (EQADC_IDCR2)

The EQADC Interrupt Control Registers (EQADC_IDCR) contain bits to enable the generation of interrupt or DMA requests when the corresponding flag bits are set in [EQADC FIFO and Interrupt Status Register \(EQADC_FISR_n\)](#).

Address: 0h base + 68h offset = 68h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	NCIE4	TORIE4	PIE4	EOQIE4	CFUIE4	0	CFIE4	CFIS4	0				RFOIE4	0	RFDE4	RFDS4
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	NCIE5	TORIE5	PIE5	EOQIE5	CFUIE5	0	CFIE5	CFIS5	0				RFOIE5	0	RFDE5	RFDS5
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

EQADC_IDCR2 field descriptions

Field	Description
0 NCIE4	Non-Coherency Interrupt Enable 4 NCIE_x enables the EQADC to generate an interrupt request when the corresponding NCF_x in EQADC FIFO and Interrupt Status Register (EQADC_FISR_n) is asserted. 0 Disable non-coherency interrupt request. 1 Enable non-coherency interrupt request.
1 TORIE4	Trigger Overrun Interrupt Enable 4 TORIE_x enables the EQADC to generate an interrupt request when the corresponding TORF_x in EQADC FIFO and Interrupt Status Register (EQADC_FISR_n) is asserted. Apart from generating an independent interrupt request for a CFIFOF_x Trigger Overrun event, the EQADC also provides a combined interrupt at which the Result FIFO Overflow Interrupt, the Command FIFO Underflow Interrupt, and the Command FIFO Trigger Overrun Interrupt requests of ALL CFIFOs are ORED. When RFOIE_x, CFUIE_x, and TORIE_x are all asserted, this combined interrupt request is asserted whenever one of the following 18 flags becomes asserted: RFOF_x, CFUF_x, and TORF_x (assuming that all interrupts are enabled). See EQADC DMA/interrupt request for details. 0 Disable trigger overrun interrupt request. 1 Enable trigger overrun interrupt request.
2 PIE4	Pause Interrupt Enable 4

Table continues on the next page...

EQADC_IDCR2 field descriptions (continued)

Field	Description
	PIEx enables the EQADC to generate an interrupt request when the corresponding PFx in EQADC FIFO and Interrupt Status Register (EQADC_FISRn) is asserted. 0 Disable pause interrupt request. 1 Enable pause interrupt request.
3 EOQIE4	End of Queue Interrupt Enable 4 EOQIEx enables the EQADC to generate an interrupt request when the corresponding EOQFx in EQADC FIFO and Interrupt Status Register (EQADC_FISRn) is asserted. 0 Disable End of Queue interrupt request. 1 Enable End of Queue interrupt request.
4 CFUIE4	CFIFO Underflow Interrupt Enable 4 CFUIEx enables the EQADC to generate an interrupt request when the corresponding CFUFx in EQADC FIFO and Interrupt Status Register (EQADC_FISRn) is asserted. Apart from generating an independent interrupt request for a CFIFOx underflow event, the EQADC also provides a combined interrupt at which the Result FIFO Overflow Interrupt, the Command FIFO Underflow Interrupt, and the Command FIFO Trigger Overrun Interrupt requests of ALL CFIFOs are ORed. When RFOIEx, CFUIEx, and TORIEx are all asserted, this combined interrupt request is asserted whenever one of the following 18 flags becomes asserted: RFOFx, CFUFx, and TORFx (assuming that all interrupts are enabled). See EQADC DMA/interrupt request for details. 0 Disable Underflow Interrupt request. 1 Enable Underflow Interrupt request.
5 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
6 CFFE4	CFIFO Fill Enable 4 CFFEx enables the EQADC to generate an interrupt request (CFFSx is negated) or DMA request (CFFSx is asserted) when CFFFx in EQADC FIFO and Interrupt Status Register (EQADC_FISRn) is asserted. NOTE: CFFEx must not be negated while a DMA transaction is in progress. 0 Disable CFIFO Fill DMA or Interrupt request. 1 Enable CFIFO Fill DMA or Interrupt request.
7 CFFS4	CFIFO Fill Select 4 CFFSx selects if a DMA or interrupt request is generated when CFFFx in EQADC FIFO and Interrupt Status Register (EQADC_FISRn) is asserted. If CFFEx is asserted, the EQADC generates an interrupt request when CFFSx is negated, or it generates a DMA request if CFFSx is asserted. NOTE: CFFSx must not be negated while a DMA transaction is in progress. 0 Generate interrupt request to move data from the system memory to CFIFOx. 1 Generate DMA request to move data from the system memory to CFIFOx.
8–11 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
12 RFOIE4	RFIFO Overflow Interrupt Enable 4 RFOIEx enables the EQADC to generate an interrupt request when the corresponding RFOFx in EQADC FIFO and Interrupt Status Register (EQADC_FISRn) is asserted.

Table continues on the next page...

EQADC_IDCR2 field descriptions (continued)

Field	Description
	Apart from generating an independent interrupt request for an RFIFOx overflow event, the EQADC also provides a combined interrupt at which the Result FIFO Overflow Interrupt, the Command FIFO Underflow Interrupt, and the Command FIFO Trigger Overrun Interrupt requests of ALL CFIFOs are ORed. When RFOIEx, CFUIEx, and TORIEx are all asserted, this combined interrupt request is asserted whenever one of the following 18 flags becomes asserted: RFOFx, CFUFx, and TORFx (assuming that all interrupts are enabled). See EQADC DMA/interrupt request for details. 0 Disable Overflow Interrupt request. 1 Enable Overflow Interrupt request.
13 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
14 RFDE4	RFIFO Drain Enable 4 RFDEx enables the EQADC to generate an interrupt request (RFDSx is negated) or DMA request (RFDSx is asserted) when RFDFx in EQADC FIFO and Interrupt Status Register (EQADC_FISRn) is asserted. NOTE: RFDEx must not be negated while a DMA transaction is in progress. 0 Disable RFIFO Drain DMA or Interrupt request. 1 Enable RFIFO Drain DMA or Interrupt request.
15 RFDS4	RFIFO Drain Select 4 RFDSx selects if a DMA or interrupt request is generated when RFDFx in EQADC FIFO and Interrupt Status Register (EQADC_FISRn) is asserted. If RFDEx is asserted, the EQADC generates an interrupt request when RFDSx is negated, or it generates a DMA request when RFDSx is asserted. NOTE: RFDSx must not be negated while a DMA transaction is in progress. 0 Generate interrupt request to move data from RFIFOx to the system memory. 1 Generate DMA request to move data from RFIFOx to the system memory.
16 NCIE5	Non-Coherency Interrupt Enable 5 NCIEx enables the EQADC to generate an interrupt request when the corresponding NCFx in EQADC FIFO and Interrupt Status Register (EQADC_FISRn) is asserted. 0 Disable non-coherency interrupt request. 1 Enable non-coherency interrupt request.
17 TORIE5	Trigger Overrun Interrupt Enable 5 TORIEx enables the EQADC to generate an interrupt request when the corresponding TORFx in EQADC FIFO and Interrupt Status Register (EQADC_FISRn) is asserted. Apart from generating an independent interrupt request for a CFIFOx Trigger Overrun event, the EQADC also provides a combined interrupt at which the Result FIFO Overflow Interrupt, the Command FIFO Underflow Interrupt, and the Command FIFO Trigger Overrun Interrupt requests of ALL CFIFOs are ORed. When RFOIEx, CFUIEx, and TORIEx are all asserted, this combined interrupt request is asserted whenever one of the following 18 flags becomes asserted: RFOFx, CFUFx, and TORFx (assuming that all interrupts are enabled). See EQADC DMA/interrupt request for details. 0 Disable trigger overrun interrupt request. 1 Enable trigger overrun interrupt request.
18 PIE5	Pause Interrupt Enable 5 PIEx enables the EQADC to generate an interrupt request when the corresponding PFx in EQADC FIFO and Interrupt Status Register (EQADC_FISRn) is asserted. 0 Disable pause interrupt request.

Table continues on the next page...

EQADC_IDCR2 field descriptions (continued)

Field	Description
	1 Enable pause interrupt request.
19 EOQIE5	End of Queue Interrupt Enable 5 EOQIE5 enables the EQADC to generate an interrupt request when the corresponding EOQFx in EQADC FIFO and Interrupt Status Register (EQADC_FISRn) is asserted. 0 Disable End of Queue interrupt request. 1 Enable End of Queue interrupt request.
20 CFUIE5	CFIFO Underflow Interrupt Enable 5 CFUIE5 enables the EQADC to generate an interrupt request when the corresponding CFUFx in EQADC FIFO and Interrupt Status Register (EQADC_FISRn) is asserted. Apart from generating an independent interrupt request for a CFIFOx underflow event, the EQADC also provides a combined interrupt at which the Result FIFO Overflow Interrupt, the Command FIFO Underflow Interrupt, and the Command FIFO Trigger Overrun Interrupt requests of ALL CFIFOs are ORed. When RFOIE5, CFUIE5, and TORIE5 are all asserted, this combined interrupt request is asserted whenever one of the following 18 flags becomes asserted: RFOFx, CFUFx, and TORFx (assuming that all interrupts are enabled). See EQADC DMA/interrupt request for details. 0 Disable Underflow Interrupt request. 1 Enable Underflow Interrupt request.
21 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
22 CFFE5	CFIFO Fill Enable 5 CFFEx enables the EQADC to generate an interrupt request (CFFSx is negated) or DMA request (CFFSx is asserted) when CFFFx in EQADC FIFO and Interrupt Status Register (EQADC_FISRn) is asserted. NOTE: CFFEx must not be negated while a DMA transaction is in progress. 0 Disable CFIFO Fill DMA or Interrupt request. 1 Enable CFIFO Fill DMA or Interrupt request.
23 CFFS5	CFIFO Fill Select 5 CFFSx selects if a DMA or interrupt request is generated when CFFFx in EQADC FIFO and Interrupt Status Register (EQADC_FISRn) is asserted. If CFFEx is asserted, the EQADC generates an interrupt request when CFFSx is negated, or it generates a DMA request if CFFSx is asserted. NOTE: CFFSx must not be negated while a DMA transaction is in progress. 0 Generate interrupt request to move data from the system memory to CFIFOx. 1 Generate DMA request to move data from the system memory to CFIFOx.
24–27 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
28 RFOIE5	RFIFO Overflow Interrupt Enable 5 RFOIE5 enables the EQADC to generate an interrupt request when the corresponding RFOFx in EQADC FIFO and Interrupt Status Register (EQADC_FISRn) is asserted. Apart from generating an independent interrupt request for an RFIFOx overflow event, the EQADC also provides a combined interrupt at which the Result FIFO Overflow Interrupt, the Command FIFO Underflow Interrupt, and the Command FIFO Trigger Overrun Interrupt requests of ALL CFIFOs are ORed. When RFOIE5, CFUIE5, and TORIE5 are all asserted, this combined interrupt request is asserted whenever one of the following 18 flags becomes asserted: RFOFx, CFUFx, and TORFx (assuming that all interrupts are enabled). See EQADC DMA/interrupt request for details.

Table continues on the next page...

EQADC_IDCR2 field descriptions (continued)

Field	Description
	0 Disable Overflow Interrupt request. 1 Enable Overflow Interrupt request.
29 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
30 RFDE5	RFIFO Drain Enable 5 RFDEx enables the EQADC to generate an interrupt request (RFDSx is negated) or DMA request (RFDSx is asserted) when RFDFx in EQADC FIFO and Interrupt Status Register (EQADC_FISRn) is asserted. NOTE: RFDEx must not be negated while a DMA transaction is in progress. 0 Disable RFIFO Drain DMA or Interrupt request. 1 Enable RFIFO Drain DMA or Interrupt request.
31 RFDS5	RFIFO Drain Select 5 RFDSx selects if a DMA or interrupt request is generated when RFDFx in EQADC FIFO and Interrupt Status Register (EQADC_FISRn) is asserted. If RFDEx is asserted, the EQADC generates an interrupt request when RFDSx is negated, or it generates a DMA request when RFDSx is asserted. NOTE: RFDSx must not be negated while a DMA transaction is in progress. 0 Generate interrupt request to move data from RFIFOx to the system memory. 1 Generate DMA request to move data from RFIFOx to the system memory.

30.5.12 EQADC FIFO and Interrupt Status Register (EQADC_FISRn)

The EQADC FIFO and Interrupt Status Registers (EQADC_FISR) contain flag and status bits for each CFIFO and RFIFO pair. Write "1" to a flag bit to clear it. Writing "0" has no effect. Status bits are read only. These bits indicate the status of the FIFO itself.

Address: 0h base + 70h offset + (4d × i), where i=0d to 5d

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	NCFX	TORFX	PFX	EOQFX	CFUFx	SSSX	CFFFX	0					RFOFX	0	RFDFX	0
W																
Reset	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	CFCTRX				TNXTPTRX				RFCTRX				POPXTPTRX			
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

EQADC_FISRn field descriptions

Field	Description
0 NCFX	Non-Coherency Flag NCFx is set whenever a command sequence being transferred through CFIFOx becomes non coherent. If NCIE_x in EQADC Interrupt and DMA Control Register 0 (EQADC_IDCR0) and NCFx are asserted, an interrupt request will be generated. Write "1" to clear NCFx. Writing a "0" has no effect. For more information refer to Command sequence non-coherency detection. 0 Command sequence being transferred by CFIFOx is coherent. 1 Command sequence being transferred by CFIFOx became non-coherent.

Table continues on the next page...

EQADC_FISR_n field descriptions (continued)

Field	Description
1 TORFX	Trigger Overrun Flag for CFIFO_x TORF_x is set when trigger overrun occurs for the specified CFIFO in edge or level trigger mode. Trigger overrun occurs when an already triggered CFIFO receives an additional trigger. When TORIEx in EQADC Interrupt and DMA Control Register 0 (EQADC_IDCR0) and TORF_x are asserted, an interrupt request will be generated. Apart from generating an independent interrupt request for a CFIFO_x Trigger Overrun event, the EQADC also provides a combined interrupt at which the Result FIFO Overflow Interrupt, the Command FIFO Underflow Interrupt, and the Command FIFO Trigger Overrun Interrupt requests of ALL CFIFOs are ORed. When RFOIEx, CFUIEx, and TORIEx are all asserted, this combined interrupt request is asserted whenever one of the following 18 flags becomes asserted: RFOF_x, CFUF_x, and TORF_x (assuming that all interrupts are enabled). See EQADC DMA/interrupt request for details. Write "1" to clear the TORF_x bit. Writing a "0" has no effect. NOTE: The trigger overrun flag will not set for CFIFOs configured for software trigger mode. 0 No trigger overrun occurred. 1 Trigger overrun occurred.
2 PFX	Pause Flag x PF behavior changes according to the CFIFO trigger mode. In edge trigger mode, PFX is set when the EQADC completes the transfer of an entry with an asserted Pause bit from CFIFO_x. In level trigger mode, when CFIFO_x is in TRIGGERED status, PFX is set when CFIFO status changes from TRIGGERED due to the detection of a closed gate. An interrupt routine, generated due to the asserted PF, can be used to verify if a complete scan of the CQueue was performed. If a closed gate is detected while no command transfers are taking place, it will have immediate effect on the CFIFO status. If a closed gate is detected while a command transfer to an on-chip CBuffer is taking place, it will only affect the CFIFO status when the transfer completes. The transfer of entries bound for the on-chip ADCs is considered completed when they are stored in the appropriate CBuffer. In software trigger mode, PFX will never become asserted. If PIE_x in EQADC Interrupt and DMA Control Register 0 (EQADC_IDCR0) and PFX are asserted, an interrupt will be generated. Write "1" to clear the PFX. Writing a "0" has no effect. See Pause status for more information on the Pause Flag. NOTE: In edge trigger mode, an asserted PFX only implies that the EQADC has finished transferring a command with an asserted PAUSE bit from CFIFO_x. It does not imply that result data for the current command and for all previously transferred commands has been returned to the appropriate RFIFO. NOTE: In software or level trigger mode, when the EQADC completes the transfer of an entry from CFIFO_x with an asserted PAUSE bit, PFX will not be set and transfer of commands will continue without pausing. 0 Entry with asserted PAUSE bit was not transferred from CFIFO_x (CFIFO in edge trigger mode), or CFIFO status did not change from TRIGGERED due to detection of a closed gate (CFIFO in level trigger mode). 1 Entry with asserted PAUSE bit was transferred from CFIFO_x (CFIFO in edge trigger mode), or CFIFO status changes from TRIGGERED due to detection of a closed gate (CFIFO in level trigger mode).
3 EOQFX	End of Queue Flag x EOQF_x indicates that an entry with an asserted EOQ bit was transferred from CFIFO_x to the on-chip ADCs - see Message format in EQADC for details about command message formats. When the EQADC completes the transfer of an entry with an asserted EOQ bit from CFIFO_x, EOQF_x will be set. The transfer of entries bound for the on-chip ADCs is considered completed when they are stored in the appropriate

Table continues on the next page...

EQADC_FISR_n field descriptions (continued)

Field	Description
	CBuffer. If the EOQIE_x bit in EQADC Interrupt and DMA Control Register 0 (EQADC_IDCR0) and EOQF_x are asserted, an interrupt will be generated. Write "1" to clear the EOQF_x bit. Writing a "0" has no effect. Refer to CQueue completion status for more information on the End of Queue Flag. NOTE: An asserted EOQF_x only implies that the EQADC has finished transferring a command with an asserted EOQ bit from CFIFO_x. It does not imply that result data for the current command and for all previously transferred commands has been returned to the appropriate RFIFO. 0 Entry with asserted EOQ bit was not transferred from CFIFO_x. 1 Entry with asserted EOQ bit was transferred from CFIFO_x.
4 CFUF _x	CFIFO Underflow Flag x CFUF_x indicates an underflow event on CFIFO_x. CFUF_x is set when CFIFO_x is in TRIGGERED state and it becomes empty. No commands will be transferred from an underflowing CFIFO, nor will command transfers from lower priority CFIFOs be blocked. When CFUIE_x in EQADC Interrupt and DMA Control Register 0 (EQADC_IDCR0) and CFUF_x are both asserted, the EQADC generates an interrupt request. Apart from generating an independent interrupt request for a CFIFO_x underflow event, the EQADC also provides a combined interrupt at which the Result FIFO Overflow Interrupt, the Command FIFO Underflow Interrupt, and the Command FIFO Trigger Overrun Interrupt requests of ALL CFIFOs are ORed. When RFOIE_x, CFUIE_x, and TORIE_x are all asserted, this combined interrupt request is asserted whenever one of the following 18 flags becomes asserted: RFOF_x, CFUF_x, and TORF_x (assuming that all interrupts are enabled). See EQADC DMA/interrupt request for details. Write "1" to clear CFUF_x. Writing a "0" has no effect. 0 No CFIFO underflow event occurred. 1 A CFIFO underflow event occurred.
5 SSS _x	CFIFO Single-Scan Status Bit x An asserted SSS_x bit enables the detection of trigger events for CFIFOs programmed into single-scan level- or edge-trigger mode, and works as trigger for CFIFOs programmed into single-scan software-trigger mode. See Single-scan mode for further details. The SSS_x bit is set by writing a "1" to the SSE_x bit in EQADC CFIFO Control Register 0 (EQADC_CFCR0). The EQADC clears the SSS_x bit when a command with an asserted EOQ bit is transferred from a CFIFO in single-scan mode, when a CFIFO is in single-scan level-trigger mode and its status changes from TRIGGERED due to the detection of a closed gate, or when the value of the CFIFO operation mode (MODE_x) in EQADC CFIFO Control Register 0 (EQADC_CFCR0) is changed to disabled. Writing to SSS_x has no effect. SSS_x has no effect in continuous-scan or in disabled mode. 0 CFIFO in single-scan level- or edge-trigger mode will ignore trigger events, or CFIFO in single-scan software-trigger mode is not triggered. 1 CFIFO in single-scan level- or edge-trigger mode will detect a trigger event, or CFIFO in single-scan software-trigger mode is triggered.
6 CFFF _x	CFIFO Fill Flag x CFFF_x is set when the CFIFO_x is not full. When CFFE_x in EQADC Interrupt and DMA Control Register 0 (EQADC_IDCR0) and CFFF_x are both asserted, an interrupt or a DMA request will be generated depending on the status of the CFFS_x bit. When CFFS_x is negated (interrupt requests selected), software clears CFFF_x by writing a "1" to it. Writing a "0" has no effect. When CFFS_x is asserted (DMA requests selected), CFFF_x is automatically cleared by the EQADC when the CFIFO becomes full. NOTE: Writing "1" to CFFF_x when CFFS_x is asserted (DMA requests selected) is not allowed. NOTE: When generation of interrupt requests is selected (CFFS_x=0), CFFF_x must only be cleared in the ISR after the CFIFO_x push register is accessed.

Table continues on the next page...

EQADC_FISR_n field descriptions (continued)

Field	Description
	0 CFIFO _x is full. 1 CFIFO _x is not full.
7–11 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
12 RFOFX	RFIFO Overflow Flag x RFOFX indicates an overflow event on RFIFO_x. RFOFX is set when RFIFO_x is already full, and a new data is received from the on-chip ADCs. The RFIFO_x will not overwrite older data in the RFIFO, and the new data will be ignored. When RFOIEx in EQADC Interrupt and DMA Control Register 0 (EQADC_IDCR0) and RFOFX are both asserted, the EQADC generates an interrupt request. Apart from generating an independent interrupt request for an RFIFO_x overflow event, the EQADC also provides a combined interrupt at which the Result FIFO Overflow Interrupt, the Command FIFO Underflow Interrupt, and the Command FIFO Trigger Overrun Interrupt requests of ALL CFIFOs are ORed. When RFOIEx, CFUIEx, and TORIEx are all asserted, this combined interrupt request is asserted whenever one of the following 18 flags becomes asserted: RFOFX, CFUFx, and TORFx (assuming that all interrupts are enabled). See EQADC DMA/interrupt request for details. Write "1" to clear RFOFX. Writing a "0" has no effect. 0 No RFIFO overflow event occurred. 1 An RFIFO overflow event occurred.
13 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
14 RFDFX	RFIFO Drain Flag x RFDFx indicates if RFIFO_x has valid entries or not. RFDFx is set when the RFIFO_x has at least one valid entry in it. When RFDEx in EQADC Interrupt and DMA Control Register 0 (EQADC_IDCR0) and RFDFx are both asserted, an interrupt or a DMA request will be generated depending on the status of the RFDSx bit. When RFDSx is negated (interrupt requests selected), software clears RFDFx by writing a "1" to it. Writing a "0" has no effect. When RFDSx is asserted (DMA requests selected), RFDFx is automatically cleared by the EQADC when the RFIFO becomes empty. NOTE: Writing "1" to RFDFx when RFDSx is asserted (DMA requests selected) is not allowed. NOTE: When the generation of interrupt requests is selected (RFDSx=0), RFDFx must only be cleared in the ISR after the RFIFOx pop register is accessed. 0 RFIFO _x is empty. 1 RFIFO _x has at least one valid entry.
15 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
16–19 CFCTRX	CFIFOx Entry Counter CFCTRx indicates the number of commands stored in the CFIFO_x. When the EQADC completes transferring a piece of new data from the CFIFO_x, it decrements CFCTRx by one. Writing a word or any bytes to the corresponding EQADC CFIFO Push Register (EQADC_CFPR_n) increments CFCTRx by one. Writing any value to CFCTRx has no effect.
20–23 TNXTPTRX	CFIFOx Transfer Next Pointer TNXTPTRx indicates the index of the next entry to be removed from CFIFO_x when it completes a transfer. When TNXTPTRx is zero, it points to the entry with the smallest memory-mapped address inside CFIFO_x. TNXTPTRx is only updated when a command transfer is completed. If the maximum index number (CFIFO depth minus one) is reached, TNXTPTRx is wrapped to zero, else, it is incremented by one. For details, see CFIFO basic functionality. Writing any value to TNXTPTRx has no effect.

Table continues on the next page...

EQADC_FISR_n field descriptions (continued)

Field	Description
24–27 RFCTR _x	RFIFO_x Entry Counter RFCTR_x indicates the number of data items stored in the RFIFO_x. When the EQADC stores a piece of new data into RFIFO_x, it increments RFCTR_x by one. Reading the whole word, half-word or any bytes of the corresponding EQADC Result FIFO Pop Register (EQADC_RFPR_n) decrements RFCTR_x by one. Writing any value to RFCTR_x itself has no effect. NOTE: When the DMA request is selected ((RFDS_x=1), the decrement of RFCTR_x should be done only readings controlled by the DMA external block. Flushing of the RFIFO_x decrements the RFCTR_x to zero and can lead the DMA system to a locked state (DMA request ON).
28–31 POPNXTPTR _x	RFIFO_x Pop Next Pointer POPNXTPTR_x indicates the index of the entry that will be returned when EQADC_RFPR_x is read. When POPNXTPTR_x is zero, it points to the entry with the smallest memory-mapped address inside RFIFO_x. POPNXTPTR_x is updated when EQADC_RFPR_x is read. If the maximum index number (RFIFO depth minus one) is reached, POPNXTPTR_x is wrapped to zero, else, it is incremented by one. For details refer to RFIFO basic functionality. Writing any value to POPNXTPTR_x has no effect.

30.5.13 EQADC CFIFO Transfer Counter Register 0 (EQADC_CFTCR0)

The EQADC CFIFO Transfer Counter Registers (EQADC_CFTCR) record the number of commands transferred from a CFIFO. The EQADC_CFTCR supports the monitoring of command transfers from a CFIFO.

Address: 0h base + 90h offset = 90h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31					
R	0					TC_CF0											0					TC_CF1															
W																																					
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0					

EQADC_CFTCR0 field descriptions

Field	Description
0–4 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
5–15 TC_CF0	Transfer Counter for CFIFO0 TC_CF_x counts the number of commands which have been completely transferred from CFIFO_x. The transfer of entries bound for the on-chip ADCs is considered completed when they are stored in the appropriate CBuffer. The EQADC increments the TC_CF_x value by one after a command is transferred. TC_CF_x resets to zero after EQADC completes transferring a command with an asserted EOQ bit. Writing any value to TC_CF_x sets the counter to that written value. NOTE: If CFIFO_x is in TRIGGERED state when its MODE_x field is programmed to disabled, the exact number of entries transferred from the CFIFO until that point - TC_CF_x - is only known after the CFIFO status changes to IDLE, as indicated by CFS_x. For details refer to Disabled Mode.

Table continues on the next page...

EQADC_CFTCR0 field descriptions (continued)

Field	Description
16–20 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
21–31 TC_CF1	Transfer Counter for CFIFO1 TC_CFx counts the number of commands which have been completely transferred from CFIFOx. The transfer of entries bound for the on-chip ADCs is considered completed when they are stored in the appropriate CBuffer. The EQADC increments the TC_CFx value by one after a command is transferred. TC_CFx resets to zero after EQADC completes transferring a command with an asserted EOQ bit. Writing any value to TC_CFx sets the counter to that written value. NOTE: If CFIFOx is in TRIGGERED state when its MODEx field is programmed to disabled, the exact number of entries transferred from the CFIFO until that point - TC_CFx - is only known after the CFIFO status changes to IDLE, as indicated by CFSx. For details refer to Disabled Mode .

30.5.14 EQADC CFIFO Transfer Counter Register 1 (EQADC_CFTCR1)

The EQADC CFIFO Transfer Counter Registers (EQADC_CFTCR) record the number of commands transferred from a CFIFO. The EQADC_CFTCR supports the monitoring of command transfers from a CFIFO.

Address: 0h base + 94h offset = 94h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R					0															0												
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

EQADC_CFTCR1 field descriptions

Field	Description
0–4 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
5–15 TC_CF2	Transfer Counter for CFIFO2 TC_CFx counts the number of commands which have been completely transferred from CFIFOx. The transfer of entries bound for the on-chip ADCs is considered completed when they are stored in the appropriate CBuffer. The EQADC increments the TC_CFx value by one after a command is transferred. TC_CFx resets to zero after EQADC completes transferring a command with an asserted EOQ bit. Writing any value to TC_CFx sets the counter to that written value. NOTE: If CFIFOx is in TRIGGERED state when its MODEx field is programmed to disabled, the exact number of entries transferred from the CFIFO until that point - TC_CFx - is only known after the CFIFO status changes to IDLE, as indicated by CFSx. For details refer to Disabled Mode .
16–20 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

Table continues on the next page...

EQADC_CFTCR1 field descriptions (continued)

Field	Description
21–31 TC_CF3	Transfer Counter for CFIFO3 TC_CFx counts the number of commands which have been completely transferred from CFIFOx. The transfer of entries bound for the on-chip ADCs is considered completed when they are stored in the appropriate CBuffer. The EQADC increments the TC_CFx value by one after a command is transferred. TC_CFx resets to zero after EQADC completes transferring a command with an asserted EOQ bit. Writing any value to TC_CFx sets the counter to that written value. NOTE: If CFIFOx is in TRIGGERED state when its MODEx field is programmed to disabled, the exact number of entries transferred from the CFIFO until that point - TC_CFx - is only known after the CFIFO status changes to IDLE, as indicated by CFSx. For details refer to Disabled Mode.

30.5.15 EQADC CFIFO Transfer Counter Register 2 (EQADC_CFTCR2)

The EQADC CFIFO Transfer Counter Registers (EQADC_CFTCR) record the number of commands transferred from a CFIFO. The EQADC_CFTCR supports the monitoring of command transfers from a CFIFO.

Address: 0h base + 98h offset = 98h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0					TC_CF4												0				TC_CF5										
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

EQADC_CFTCR2 field descriptions

Field	Description
0–4 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
5–15 TC_CF4	Transfer Counter for CFIFO4 TC_CFx counts the number of commands which have been completely transferred from CFIFOx. The transfer of entries bound for the on-chip ADCs is considered completed when they are stored in the appropriate CBuffer. The EQADC increments the TC_CFx value by one after a command is transferred. TC_CFx resets to zero after EQADC completes transferring a command with an asserted EOQ bit. Writing any value to TC_CFx sets the counter to that written value. NOTE: If CFIFOx is in TRIGGERED state when its MODEx field is programmed to disabled, the exact number of entries transferred from the CFIFO until that point - TC_CFx - is only known after the CFIFO status changes to IDLE, as indicated by CFSx. For details refer to Disabled Mode.
16–20 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
21–31 TC_CF5	Transfer Counter for CFIFO5

Table continues on the next page...

EQADC_CFTCR2 field descriptions (continued)

Field	Description
	TC_CFx counts the number of commands which have been completely transferred from CFIFOfx. The transfer of entries bound for the on-chip ADCs is considered completed when they are stored in the appropriate CBuffer. The EQADC increments the TC_CFx value by one after a command is transferred. TC_CFx resets to zero after EQADC completes transferring a command with an asserted EOQ bit. Writing any value to TC_CFx sets the counter to that written value. NOTE: If CFIFOfx is in TRIGGERED state when its MODEx field is programmed to disabled, the exact number of entries transferred from the CFIFO until that point - TC_CFx - is only known after the CFIFO status changes to IDLE, as indicated by CFSx. For details refer to Disabled Mode.

30.5.16 EQADC CFIFO Status Snapshot Register 0 (EQADC_CFSSR0)

The EQADC CFIFO Status Snapshot Registers (EQADC_CFSSR) contain status fields to track the operation status of each CFIFO and the transfer counter of the last CFIFO to initiate a command transfer to the internal CBuffers. EQADC_CFSSR0-1 are related to the on-chip CBuffers (CBuffer0-1). All fields of a particular EQADC_CFSSR register are captured at the beginning of a command transfer to the CBuffer associated with that register. Note that captured status register values are associated with previous command transfer. This means that the CFSSR registers capture the status registers before the status registers change because of the transfer of the current command that is about to be popped from the CFIFO. The EQADC_CFSSR registers are read only. Writing to the EQADC_CFSSR registers has no effect.

Address: 0h base + A0h offset = A0h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	CFS0_TCB0		CFS1_TCB0		CFS2_TCB0		CFS3_TCB0		CFS4_TCB0		CFS5_TCB0		0			
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	LCFTCB0					TC_LCFTCB0									
W																
Reset	0	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0

EQADC_CFSSR0 field descriptions

Field	Description
0-1 CFS0_TCB0	CFIFO Status at Transfer to CBuffer0 CFSx_TCBn indicates the CFIFOfx status of previously completed command transfer. CFSx_TCBn is a copy of the corresponding CFSx in EQADC CFIFO Status Register (EQADC_CFSR), captured at the time a current command transfer to CBuffern is initiated.

Table continues on the next page...

EQADC_CFSSR0 field descriptions (continued)

Field	Description
2–3 CFS1_TCB0	CFIFO Status at Transfer to CBuffer0 CFSx_TCBn indicates the CFIFOx status of previously completed command transfer. CFSx_TCBn is a copy of the corresponding CFSx in EQADC CFIFO Status Register (EQADC_CFSR) , captured at the time a current command transfer to CBuffern is initiated.
4–5 CFS2_TCB0	CFIFO Status at Transfer to CBuffer0 CFSx_TCBn indicates the CFIFOx status of previously completed command transfer. CFSx_TCBn is a copy of the corresponding CFSx in EQADC CFIFO Status Register (EQADC_CFSR) , captured at the time a current command transfer to CBuffern is initiated.
6–7 CFS3_TCB0	CFIFO Status at Transfer to CBuffer0 CFSx_TCBn indicates the CFIFOx status of previously completed command transfer. CFSx_TCBn is a copy of the corresponding CFSx in EQADC CFIFO Status Register (EQADC_CFSR) , captured at the time a current command transfer to CBuffern is initiated.
8–9 CFS4_TCB0	CFIFO Status at Transfer to CBuffer0 CFSx_TCBn indicates the CFIFOx status of previously completed command transfer. CFSx_TCBn is a copy of the corresponding CFSx in EQADC CFIFO Status Register (EQADC_CFSR) , captured at the time a current command transfer to CBuffern is initiated.
10–11 CFS5_TCB0	CFIFO Status at Transfer to CBuffer0 CFSx_TCBn indicates the CFIFOx status of previously completed command transfer. CFSx_TCBn is a copy of the corresponding CFSx in EQADC CFIFO Status Register (EQADC_CFSR) , captured at the time a current command transfer to CBuffern is initiated.
12–16 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
17–20 LCFTCB0	Last CFIFO to Transfer to CBuffer0 LCFTCBn holds the CFIFO number to have completed a previous command transfer to CBuffern. 0000 Last command was transferred from CFIFO0 0001 Last command was transferred from CFIFO1 0010 Last command was transferred from CFIFO2 0011 Last command was transferred from CFIFO3 0100 Last command was transferred from CFIFO4 0101 Last command was transferred from CFIFO5 0110 Reserved 0111 Reserved 1000 Reserved 1001 Reserved 1010 Reserved 1011 Reserved 1100 Reserved 1101 Reserved 1110 Reserved 1111 No command was transferred to CBuffern
21–31 TC_LCFTCB0	Transfer Counter for Last CFIFO to transfer commands to CBuffer0

Table continues on the next page...

EQADC_CFSSR0 field descriptions (continued)

Field	Description
	TC_LCFTCBn indicates the number of commands which have been completely transferred from CFIFOx when a current command transfer from CFIFOx to CBuffern is initiated. TC_LCFTCBn is a copy of the corresponding TC_CFx in EQADC CFIFO Transfer Counter Register 0 (EQADC_CFTCR0) , captured at the time a current command transfer from CFIFOx to CBuffern is initiated. This field has no meaning when LCFTCBn is 0b1111.

30.5.17 EQADC CFIFO Status Snapshot Register 1 (EQADC_CFSSR1)

The EQADC CFIFO Status Snapshot Registers (EQADC_CFSSR) contain status fields to track the operation status of each CFIFO and the transfer counter of the last CFIFO to initiate a command transfer to the internal CBuffers. EQADC_CFSSR0-1 are related to the on-chip CBuffers (CBuffer0-1). All fields of a particular EQADC_CFSSR register are captured at the beginning of a command transfer to the CBuffer associated with that register. Note that captured status register values are associated with previous command transfer. This means that the CFSSR registers capture the status registers before the status registers change because of the transfer of the current command that is about to be popped from the CFIFO. The EQADC_CFSSR registers are read only. Writing to the EQADC_CFSSR registers has no effect.

Address: 0h base + A4h offset = A4h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	CFS0_TCB1	CFS1_TCB1	CFS2_TCB1	CFS3_TCB1	CFS4_TCB1	CFS5_TCB1	0									
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	LCFTCB1				TC_LCFTCB1										
W																
Reset	0	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0

EQADC_CFSSR1 field descriptions

Field	Description
0–1 CFS0_TCB1	CFIFO Status at Transfer to CBuffer1 CFSx_TCBn indicates the CFIFOx status of previously completed command transfer. CFSx_TCBn is a copy of the corresponding CFSx in EQADC CFIFO Status Register (EQADC_CFSR) , captured at the time a current command transfer to CBuffern is initiated.
2–3 CFS1_TCB1	CFIFO Status at Transfer to CBuffer1

Table continues on the next page...

EQADC_CFSSR1 field descriptions (continued)

Field	Description
	CFSx_TCBn indicates the CFIFOx status of previously completed command transfer. CFSx_TCBn is a copy of the corresponding CFSx in EQADC CFIFO Status Register (EQADC_CFSR) , captured at the time a current command transfer to CBuffern is initiated.
4–5 CFS2_TCB1	CFIFO Status at Transfer to CBuffer1 CFSx_TCBn indicates the CFIFOx status of previously completed command transfer. CFSx_TCBn is a copy of the corresponding CFSx in EQADC CFIFO Status Register (EQADC_CFSR) , captured at the time a current command transfer to CBuffern is initiated.
6–7 CFS3_TCB1	CFIFO Status at Transfer to CBuffer1 CFSx_TCBn indicates the CFIFOx status of previously completed command transfer. CFSx_TCBn is a copy of the corresponding CFSx in EQADC CFIFO Status Register (EQADC_CFSR) , captured at the time a current command transfer to CBuffern is initiated.
8–9 CFS4_TCB1	CFIFO Status at Transfer to CBuffer1 CFSx_TCBn indicates the CFIFOx status of previously completed command transfer. CFSx_TCBn is a copy of the corresponding CFSx in EQADC CFIFO Status Register (EQADC_CFSR) , captured at the time a current command transfer to CBuffern is initiated.
10–11 CFS5_TCB1	CFIFO Status at Transfer to CBuffer1 CFSx_TCBn indicates the CFIFOx status of previously completed command transfer. CFSx_TCBn is a copy of the corresponding CFSx in EQADC CFIFO Status Register (EQADC_CFSR) , captured at the time a current command transfer to CBuffern is initiated.
12–16 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
17–20 LCFTCB1	Last CFIFO to Transfer to CBuffer1 LCFTCBn holds the CFIFO number to have completed a previous command transfer to CBuffern. 0000 Last command was transferred from CFIFO0 0001 Last command was transferred from CFIFO1 0010 Last command was transferred from CFIFO2 0011 Last command was transferred from CFIFO3 0100 Last command was transferred from CFIFO4 0101 Last command was transferred from CFIFO5 0110 Reserved 0111 Reserved 1000 Reserved 1001 Reserved 1010 Reserved 1011 Reserved 1100 Reserved 1101 Reserved 1110 Reserved 1111 No command was transferred to CBuffern
21–31 TC_LCFTCB1	Transfer Counter for Last CFIFO to transfer commands to CBuffer1 TC_LCFTCBn indicates the number of commands which have been completely transferred from CFIFOx when a current command transfer from CFIFOx to CBuffern is initiated. TC_LCFTCBn is a copy of the corresponding TC_CFx in EQADC CFIFO Transfer Counter Register 0 (EQADC_CFTCR0) , captured at the time a current command transfer from CFIFOx to CBuffern is initiated. This field has no meaning when LCFTCBn is 0b1111.

30.5.18 EQADC CFIFO Status Register (EQADC_CFSR)

The EQADC CFIFO Status Register (EQADC_CFSR) contains the current CFIFO status. The EQADC_CFSR registers is read only. Writing to the EQADC_CFSR register has no effect.

Table 30-43. Current CFIFO Status

CFIFO Status	Field Value	Explanation
IDLE	0b00	CFIFO is disabled. CFIFO is in single-scan edge or level trigger mode and does not have SSS asserted. EQADC completed the transfer of the last entry of the CQueue in single-scan mode.
Reserved	0b01	Not applicable.
WAITING FOR TRIGGER	0b10	CFIFO Mode is modified to continuous-scan edge or level trigger mode. CFIFO Mode is modified to single-scan edge or level trigger mode and SSS is asserted. CFIFO Mode is modified to single-scan software trigger mode and SSS is negated. CFIFO is paused. EQADC transferred the last entry of the queue in continuous-scan edge trigger mode.
TRIGGERED	0b11	CFIFO is triggered

Address: 0h base + ACh offset = ACh

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	CFS0		CFS1		CFS2		CFS3		CFS4		CFS5		0																			
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

EQADC_CFSR field descriptions

Field	Description
0–1 CFS0	CFIFO Status CFSx indicates the current status of CFIFOx. See Table 30-43 for more information on CFIFO status.
2–3 CFS1	CFIFO Status CFSx indicates the current status of CFIFOx. See Table 30-43 for more information on CFIFO status.
4–5 CFS2	CFIFO Status CFSx indicates the current status of CFIFOx. See Table 30-43 for more information on CFIFO status.
6–7 CFS3	CFIFO Status CFSx indicates the current status of CFIFOx. See Table 30-43 for more information on CFIFO status.

Table continues on the next page...

EQADC_CFSR field descriptions (continued)

Field	Description
8–9 CFS4	CFIFO Status CFSx indicates the current status of CFIFOx. See Table 30-43 for more information on CFIFO status.
10–11 CFS5	CFIFO Status CFSx indicates the current status of CFIFOx. See Table 30-43 for more information on CFIFO status.
12–31 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

30.5.19 EQADC STAC Bus Client Configuration Register (EQADC_STACCCR)

The EQADC STAC Bus Client Configuration Register (EQADC_STACCCR) contains bits used to control which time slots the EQADC selects to obtain pre-defined external time bases.

Table 30-45. STAC Bus timebase bits selection

STACBSm[0:3]	Selected Bits
0b0000	TBASEm[0:15]
0b0001	TBASEm[1:16]
0b0010	TBASEm[2:17]
0b0011	TBASEm[3:18]
0b0100	TBASEm[4:19]
0b0101	TBASEm[5:20]
0b0110	TBASEm[6:21]
0b0111	TBASEm[7:22]
0b1000	TBASEm[8:23]
Others	Reserved

Address: 0h base + D0h offset = D0h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0																STACBS2				SRV2				STACBS1				SRV1			
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

EQADC_STACCCR field descriptions

Field	Description
0–15 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

Table continues on the next page...

EQADC_STACCCR field descriptions (continued)

Field	Description
16–19 STACBS2	STAC Bus Timebase Bits Selection 2 The STACBSm field selects 16 bits from the total of 24 bits that are received from the STAC Bus interface as described in Table 30-45 . Consider TBASEm[0:23] the selected time base from slot SRVm.
20–23 SRV2	STAC Bus Server Data Slot Selector 2 The field SRVm indicates the slot number that contains the desired time base value sent by the STAC Bus server. It is possible to have up to 16 different sources to be selected.
24–27 STACBS1	STAC Bus Timebase Bits Selection 1 The STACBSm field selects 16 bits from the total of 24 bits that are received from the STAC Bus interface as described in Table 30-45 . Consider TBASEm[0:23] the selected time base from slot SRVm.
28–31 SRV1	STAC Bus Server Data Slot Selector 1 The field SRVm indicates the slot number that contains the desired time base value sent by the STAC Bus server. It is possible to have up to 16 different sources to be selected.

30.5.20 EQADC CFIFO0 Register (EQADC_CF0Rn)

The EQADC CFIFO Registers (EQADC_CFxRw) (x=0, .., 5; w=0, .., 3) provide visibility of the contents of a CFIFO for debugging purposes. Each CFIFO has four registers which are uniquely mapped to its four 32-bit entries. See [EQADC command FIFOs](#) for more information on CFIFOs. These registers are read only. Data written to these registers is ignored.

Address: 0h base + 100h offset + (4d × i), where i=0d to 3d

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	CFIFO0_DATAW																															
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	

EQADC_CF0Rn field descriptions

Field	Description
0–31 CFIFO0_DATAW	CFIFO0 Data w (w = 0, .., 3) Reading CFIFOx_DATAw returns the value stored on the wth entry of CFIFOx. Each CFIFO is composed of four 32-bit entries, with register 0 being mapped to the one with the smallest memory mapped address.

30.5.21 EQADC CFIFO0 Extension Register (EQADC_CF0ER_n)

The EQADC CFIFO0 Extension Registers (EQADC_CF0ER_w) (w=0, ..., 3) provide visibility of the contents of the extended portion of CFIFO0 for debugging purposes. There are four registers which are uniquely mapped to its four 32-bit entries. See [EQADC command FIFOs](#) for more information on CFIFOs. These registers are read only. Data written to these registers is ignored.

Address: 0h base + 110h offset + (4d × i), where i=0d to 3d

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	CFIFO0_EDATAW																															
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

EQADC_CF0ER_n field descriptions

Field	Description
0–31 CFIFO0_ EDATAW	CFIFO0 Data w (w = 0, ..., 3) Reading CFIFO0_DATAw returns the value stored on the wth entry of CFIFO0. Each CFIFO is composed of four 32-bit entries, with register 0 being mapped to the one with the smallest memory mapped address.

30.5.22 EQADC CFIFO1 Register (EQADC_CF1R_n)

The EQADC CFIFO Registers (EQADC_CF_xR_w) (x=0, ..., 5; w=0, ..., 3) provide visibility of the contents of a CFIFO for debugging purposes. Each CFIFO has four registers which are uniquely mapped to its four 32-bit entries. See [EQADC command FIFOs](#) for more information on CFIFOs. These registers are read only. Data written to these registers is ignored.

Address: 0h base + 140h offset + (4d × i), where i=0d to 3d

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	CFIFO1_DATAW																															
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

EQADC_CF1Rn field descriptions

Field	Description
0–31 CFIFO1_DATAw	CFIFO1 Data w (w = 0, ..., 3) Reading CFIFOx_DATAw returns the value stored on the wth entry of CFIFOx. Each CFIFO is composed of four 32-bit entries, with register 0 being mapped to the one with the smallest memory mapped address.

30.5.23 EQADC CFIFO2 Register (EQADC_CF2Rn)

The EQADC CFIFO Registers (EQADC_CFxRw) (x=0, ..., 5; w=0, ..., 3) provide visibility of the contents of a CFIFO for debugging purposes. Each CFIFO has four registers which are uniquely mapped to its four 32-bit entries. See [EQADC command FIFOs](#) for more information on CFIFOs. These registers are read only. Data written to these registers is ignored.

Address: 0h base + 180h offset + (4d × i), where i=0d to 3d

[illegible]

EQADC_CF2Rn field descriptions

Field	Description
0–31 CFIFO2_DATAw	CFIFO2 Data w (w = 0, ..., 3) Reading CFIFOx_DATAw returns the value stored on the wth entry of CFIFOx. Each CFIFO is composed of four 32-bit entries, with register 0 being mapped to the one with the smallest memory mapped address.

30.5.24 EQADC CFIFO3 Register (EQADC_CF3Rn)

The EQADC CFIFO Registers (EQADC_CFxRw) ($x=0, \dots, 5$; $w=0, \dots, 3$) provide visibility of the contents of a CFIFO for debugging purposes. Each CFIFO has four registers which are uniquely mapped to its four 32-bit entries. See [EQADC command FIFOs](#) for more information on CFIFOs. These registers are read only. Data written to these registers is ignored.

Address: $0h \text{ base} + 1C0h \text{ offset} + (4d \times i)$, where $i=0d$ to $3d$

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	CFIFO3_DATAW																															
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

EQADC_CF3Rn field descriptions

Field	Description
0–31 CFIFO3_DATAW	CFIFO3 Data w ($w = 0, \dots, 3$) Reading CFIFOx_DATAw returns the value stored on the wth entry of CFIFOx. Each CFIFO is composed of four 32-bit entries, with register 0 being mapped to the one with the smallest memory mapped address.

30.5.25 EQADC CFIFO4 Register (EQADC_CF4Rn)

The EQADC CFIFO Registers (EQADC_CFxRw) ($x=0, \dots, 5$; $w=0, \dots, 3$) provide visibility of the contents of a CFIFO for debugging purposes. Each CFIFO has four registers which are uniquely mapped to its four 32-bit entries. See [EQADC command FIFOs](#) for more information on CFIFOs. These registers are read only. Data written to these registers is ignored.

Address: $0h \text{ base} + 200h \text{ offset} + (4d \times i)$, where $i=0d$ to $3d$

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	CFIFO4_DATAW																															
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

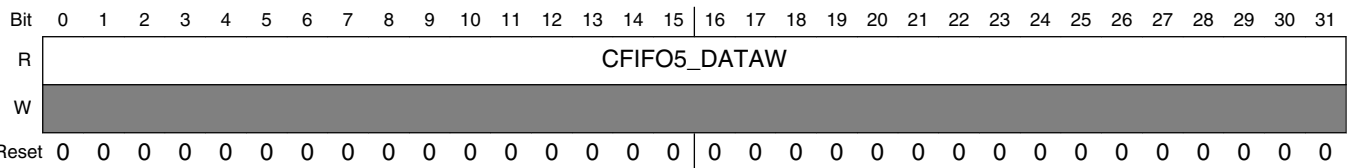
EQADC_CF4Rn field descriptions

Field	Description
0–31 CFIFO4_DATAW	CFIFO4 Data w (w = 0, ..., 3) Reading CFIFOx_DATAw returns the value stored on the wth entry of CFIFOx. Each CFIFO is composed of four 32-bit entries, with register 0 being mapped to the one with the smallest memory mapped address.

30.5.26 EQADC CFIFO5 Register (EQADC_CF5Rn)

The EQADC CFIFO Registers (EQADC_CFxRw) (x=0, ..., 5; w=0, ..., 3) provide visibility of the contents of a CFIFO for debugging purposes. Each CFIFO has four registers which are uniquely mapped to its four 32-bit entries. See [EQADC command FIFOs](#) for more information on CFIFOs. These registers are read only. Data written to these registers is ignored.

Address: 0h base + 240h offset + (4d × i), where i=0d to 3d



EQADC_CF5Rn field descriptions

Field	Description
0–31 CFIFO5_DATAW	CFIFO5 Data w (w = 0, ..., 3) Reading CFIFOx_DATAw returns the value stored on the wth entry of CFIFOx. Each CFIFO is composed of four 32-bit entries, with register 0 being mapped to the one with the smallest memory mapped address.

30.5.27 EQADC RFIFO0 Register (EQADC_RF0Rn)

The EQADC RFIFO Registers (EQADC_RFxRw) ($x=0, \dots, 5$; $w=0, \dots, 3$) provide visibility of the contents of a RFIFO for debugging purposes. Each RFIFO has four registers which are uniquely mapped to its four 16-bit entries. See [EQADC result FIFOs](#) for more information on RFIFOs. These registers are read only. Data written to these registers is ignored.

Address: $0h \text{ base} + 300h \text{ offset} + (4d \times i)$, where $i=0d$ to $3d$

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0																RFIFO0_DATAW															
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	

EQADC_RF0Rn field descriptions

Field	Description
0–15 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
16–31 RFIFO0_DATAW	RFIFO0 Data w ($w = 0, \dots, 3$) Reading RFIFOx_DATAw returns the value stored on the wth entry of RFIFOx. Each RFIFO is composed of four 16-bit entries, with register 0 being mapped to the one with the smallest memory mapped address.

30.5.28 EQADC RFIFO1 Register (EQADC_RF1Rn)

The EQADC RFIFO Registers (EQADC_RFxRw) ($x=0, \dots, 5$; $w=0, \dots, 3$) provide visibility of the contents of a RFIFO for debugging purposes. Each RFIFO has four registers which are uniquely mapped to its four 16-bit entries. See [EQADC result FIFOs](#) for more information on RFIFOs. These registers are read only. Data written to these registers is ignored.

Address: $0h \text{ base} + 340h \text{ offset} + (4d \times i)$, where $i=0d$ to $3d$

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0																RFIFO1_DATAW															
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	

EQADC_RF1Rn field descriptions

Field	Description
0–15 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
16–31 RFIFO1_DATAW	RFIFO1 Data w (w = 0, ..., 3) Reading RFIFOx_DATAw returns the value stored on the wth entry of RFIFOx. Each RFIFO is composed of four 16-bit entries, with register 0 being mapped to the one with the smallest memory mapped address.

30.5.29 EQADC RFIFO2 Register (EQADC_RF2Rn)

The EQADC RFIFO Registers (EQADC_RFxRw) (x=0, ..., 5; w=0, ..., 3) provide visibility of the contents of a RFIFO for debugging purposes. Each RFIFO has four registers which are uniquely mapped to its four 16-bit entries. See [EQADC result FIFOs](#) for more information on RFIFOs. These registers are read only. Data written to these registers is ignored.

Address: 0h base + 380h offset + (4d × i), where i=0d to 3d

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0																RFIFO2_DATAW															
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	

EQADC_RF2Rn field descriptions

Field	Description
0–15 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
16–31 RFIFO2_DATAW	RFIFO2 Data w (w = 0, ..., 3) Reading RFIFOx_DATAw returns the value stored on the wth entry of RFIFOx. Each RFIFO is composed of four 16-bit entries, with register 0 being mapped to the one with the smallest memory mapped address.

30.5.30 EQADC RFIFO3 Register (EQADC_RF3Rn)

The EQADC RFIFO Registers (EQADC_RFxRw) ($x=0, \dots, 5$; $w=0, \dots, 3$) provide visibility of the contents of a RFIFO for debugging purposes. Each RFIFO has four registers which are uniquely mapped to its four 16-bit entries. See [EQADC result FIFOs](#) for more information on RFIFOs. These registers are read only. Data written to these registers is ignored.

Address: $0h \text{ base} + 3C0h \text{ offset} + (4d \times i)$, where $i=0d$ to $3d$

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0																RFIFO3_DATAW															
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	

EQADC_RF3Rn field descriptions

Field	Description
0–15 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
16–31 RFIFO3_DATAW	RFIFO3 Data w ($w = 0, \dots, 3$) Reading RFIFOx_DATAw returns the value stored on the wth entry of RFIFOx. Each RFIFO is composed of four 16-bit entries, with register 0 being mapped to the one with the smallest memory mapped address.

30.5.31 EQADC RFIFO4 Register (EQADC_RF4Rn)

The EQADC RFIFO Registers (EQADC_RFxRw) ($x=0, \dots, 5$; $w=0, \dots, 3$) provide visibility of the contents of a RFIFO for debugging purposes. Each RFIFO has four registers which are uniquely mapped to its four 16-bit entries. See [EQADC result FIFOs](#) for more information on RFIFOs. These registers are read only. Data written to these registers is ignored.

Address: $0h \text{ base} + 400h \text{ offset} + (4d \times i)$, where $i=0d$ to $3d$

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0																RFIFO4_DATAW															
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	

EQADC_RF4Rn field descriptions

Field	Description
0–15 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
16–31 RFIFO4_DATAW	RFIFO4 Data w (w = 0, ..., 3) Reading RFIFOx_DATAw returns the value stored on the wth entry of RFIFOx. Each RFIFO is composed of four 16-bit entries, with register 0 being mapped to the one with the smallest memory mapped address.

30.5.32 EQADC RFIFO5 Register (EQADC_RF5Rn)

The EQADC RFIFO Registers (EQADC_RFxRw) (x=0, ..., 5; w=0, ..., 3) provide visibility of the contents of a RFIFO for debugging purposes. Each RFIFO has four registers which are uniquely mapped to its four 16-bit entries. See [EQADC result FIFOs](#) for more information on RFIFOs. These registers are read only. Data written to these registers is ignored.

Address: 0h base + 440h offset + (4d × i), where i=0d to 3d

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0																RFIFO5_DATAW															
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	

EQADC_RF5Rn field descriptions

Field	Description
0–15 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
16–31 RFIFO5_DATAW	RFIFO5 Data w (w = 0, ..., 3) Reading RFIFOx_DATAw returns the value stored on the wth entry of RFIFOx. Each RFIFO is composed of four 16-bit entries, with register 0 being mapped to the one with the smallest memory mapped address.

30.5.33 On-chip ADC registers

This section describes a list of registers that control on-chip ADC operation. The ADC registers are not part of the CPU accessible memory map. These registers can only be accessed indirectly through configuration commands. There are non memory mapped registers that are individual per ADC, plus other registers that are shared by both ADCs. The address, usage, and access privilege of each register is shown in [Table 30-111](#). Data written to or read from reserved areas of the memory map is undefined.

Their assigned addresses are the values used to set the ADC_REG_ADDRESS field of the read/write configurations commands bound for the on-chip ADCs. These are half-word addresses. Further, the following restrictions apply when accessing these registers:

- Registers ADC0_CR, ADC0_GCCR, ADC0_OCCR, ADC0_AGR1/2 and ADC0_AOR1/2 can only be accessed by configuration commands sent to CBuffer0.
- Registers ADC1_CR, ADC1_GCCR, ADC1_OCCR, ADC1_AGR1/2 and ADC1_AOR1/2 can only be accessed by configuration commands sent to CBuffer1.
- Registers ADC_TSCR, ADC_TBCR, ADC_ACR1-14, ADC_EACR1-14 and ADC_PUDCR0-7 can be accessed by configuration commands sent to CBuffer0 or to CBuffer1. A data write to any of these registers through a configuration command sent to CBuffer0 will write the same memory location as when writing to it through a configuration command sent to CBuffer1.

Note

Simultaneous write accesses from CBuffer0 and CBuffer1 to any of the shared registers are not allowed.

Table 30-111. On-Chip ADC Memory Map

ADC address	Use	Access
0x00	ADC0/ADC1 ¹ Conversion Command for Standard Configuration (See Conversion command format for the standard configuration)	Write
0x01	ADC0/ADC1 Configuration Control Register (ADC0_CR, ADC1_CR)	Write/Read
0x02	Time Stamp Control Register (ADC_TSCR)	Write/Read
0x03	Time Base Counter Register (ADC_TBCR)	Write/Read
0x04	ADC0/ADC1 Gain Calibration Constant Register (ADC0_GCCR, ADC1_GCCR)	Write/Read
0x05	ADC0/ADC1 Offset Calibration Constant Register (ADC0_OCCR, ADC1_OCCR)	Write/Read
0x06- 0x07	Reserved	-
0x08	ADC0/ADC1 Conversion Command for Alternate Configuration 1 (See Conversion command format for alternate configurations)	Write
0x09	ADC0/ADC1 Conversion Command for Alternate Configuration 2 (See Conversion command format for alternate configurations)	Write
0x0A	ADC0/ADC1 Conversion Command for Alternate Configuration 3 (See Conversion command format for alternate configurations)	Write
0x0B	ADC0/ADC1 Conversion Command for Alternate Configuration 4 (See Conversion command format for alternate configurations)	Write
0x0C	ADC0/ADC1 Conversion Command for Alternate Configuration 5 (See Conversion command format for alternate configurations)	Write
0x0D	ADC0/ADC1 Conversion Command for Alternate Configuration 6 (See Conversion command format for alternate configurations)	Write

Table continues on the next page...

Table 30-111. On-Chip ADC Memory Map (continued)

ADC address	Use	Access
0x0E	ADC0/ADC1 Conversion Command for Alternate Configuration 7 (See Conversion command format for alternate configurations)	Write
0x0F	ADC0/ADC1 Conversion Command for Alternate Configuration 8 (See Conversion command format for alternate configurations)	Write
0x10	ADC0/ADC1 Conversion Command for Alternate Configuration 9 (See Conversion command format for alternate configurations)	Write
0x11	ADC0/ADC1 Conversion Command for Alternate Configuration 10 (See Conversion command format for alternate configurations)	Write
0x12	ADC0/ADC1 Conversion Command for Alternate Configuration 11 (See Conversion command format for alternate configurations)	Write
0x13	ADC0/ADC1 Conversion Command for Alternate Configuration 12 (See Conversion command format for alternate configurations)	Write
0x14	ADC0/ADC1 Conversion Command for Alternate Configuration 13 (See Conversion command format for alternate configurations)	Write
0x15	ADC0/ADC1 Conversion Command for Alternate Configuration 14 (See Conversion command format for alternate configurations)	Write
0x16-0x2F	Reserved	-
0x30	Alternate Configuration 1 Control Register (ADC_ACR1)	Write/Read
0x31	ADC0/ADC1 Alternate Gain 1 Register (ADC0_AGR1, ADC1_AGR1)	Write/Read
0x32	ADC0/ADC1 Alternate Offset 1 Register (ADC0_AOR1, ADC1_AOR1)	Write/Read
0x33	Extended Alternate Configuration 1 Control Register (ADC_EACR1)	Write/Read
0x34	Alternate Configuration 2 Control Register (ADC_ACR2)	Write/Read
0x35	ADC0/ADC1 Alternate Gain 2 Register (ADC0_AGR2, ADC1_AGR2)	Write/Read
0x36	ADC0/ADC1 Alternate Offset 2 Register (ADC0_AOR2, ADC1_AOR2)	Write/Read
0x37	Extended Alternate Configuration 2 Control Register (ADC_EACR2)	Write/Read
0x38	Alternate Configuration 3 Control Register (ADC_ACR3)	Write/Read
0x39	Reserved	-
0x3A	Reserved	-
0x3B	Extended Alternate Configuration 3 Control Register (ADC_EACR3)	Write/Read
0x3C	Alternate Configuration 4 Control Register (ADC_ACR4)	Write/Read
0x3D	Reserved	-
0x3E	Reserved	-
0x3F	Extended Alternate Configuration 4 Control Register (ADC_EACR4)	Write/Read
0x40	Alternate Configuration 5 Control Register (ADC_ACR5)	Write/Read
0x41	Reserved	-
0x42	Reserved	-
0x43	Extended Alternate Configuration 5 Control Register (ADC_EACR5)	Write/Read
0x44	Alternate Configuration 6 Control Register (ADC_ACR6)	Write/Read
0x45	Reserved	-

Table continues on the next page...

Table 30-111. On-Chip ADC Memory Map (continued)

ADC address	Use	Access
0x46	Reserved	-
0x47	Extended Alternate Configuration 6 Control Register (ADC_EACR6)	Write/Read
0x48	Alternate Configuration 7 Control Register (ADC_ACR7)	Write/Read
0x49	Reserved	-
0x4A	Reserved	-
0x4B	Extended Alternate Configuration 7 Control Register (ADC_EACR7)	Write/Read
0x4C	Alternate Configuration 8 Control Register (ADC_ACR8)	Write/Read
0x4D	Reserved	-
0x4E	Reserved	-
0x4F	Extended Alternate Configuration 8 Control Register (ADC_EACR8)	Write/Read
0x50	Alternate Configuration 9 Control Register (ADC_ACR9)	Write/Read
0x51	Reserved	-
0x52	Reserved	-
0x53	Extended Alternate Configuration 9 Control Register (ADC_EACR9)	Write/Read
0x54	Alternate Configuration 10 Control Register (ADC_ACR10)	Write/Read
0x55	Reserved	-
0x56	Reserved	-
0x57	Extended Alternate Configuration 10 Control Register (ADC_EACR10)	Write/Read
0x58	Alternate Configuration 11 Control Register (ADC_ACR11)	Write/Read
0x59	Reserved	-
0x5A	Reserved	-
0x5B	Extended Alternate Configuration 11 Control Register (ADC_EACR11)	Write/Read
0x5C	Alternate Configuration 12 Control Register (ADC_ACR12)	Write/Read
0x5D	Reserved	-
0x5E	Reserved	-
0x5F	Extended Alternate Configuration 12 Control Register (ADC_EACR12)	Write/Read
0x60	Alternate Configuration 13 Control Register (ADC_ACR13)	Write/Read
0x61	Reserved	-
0x62	Reserved	-
0x63	Extended Alternate Configuration 13 Control Register (ADC_EACR13)	Write/Read
0x64	Alternate Configuration 14 Control Register (ADC_ACR14)	Write/Read
0x65	Reserved	-
0x66	Reserved	-
0x67	Extended Alternate Configuration 14 Control Register (ADC_EACR14)	Write/Read
0x68-0x6F	Reserved	-
0x70	Pull Up/Down Control Register0 (ADC_PUDCR0)	Write/Read

Table continues on the next page...

Table 30-111. On-Chip ADC Memory Map (continued)

ADC address	Use	Access
0x71	Pull Up/Down Control Register1 (ADC_PUDCR1)	Write/Read
0x72	Pull Up/Down Control Register2 (ADC_PUDCR2)	Write/Read
0x73	Pull Up/Down Control Register3 (ADC_PUDCR3)	Write/Read
0x74	Pull Up/Down Control Register4 (ADC_PUDCR4)	Write/Read
0x75	Pull Up/Down Control Register5 (ADC_PUDCR5)	Write/Read
0x76	Pull Up/Down Control Register6 (ADC_PUDCR6)	Write/Read
0x77	Pull Up/Down Control Register7 (ADC_PUDCR7)	Write/Read
0x78-0x97	Pull Up/Down Control Register 8 to 39 (ADC_PUDCR8 to ADC_PUDCR39) ²	Write/Read
0x98-0x9F	Reserved	-
0xA0	STAC Bus Time Base 1 Register (ADC_STACTBR1)	Read
0xA1	STAC Bus Time Base 2 Register (ADC_STACTBR2)	Read
0xA2-0xFE	Reserved	-
0xFF	Reserved for dummy write command ³	Write

1. Throughout the table, ADC0/ADC1 indicates that if the command is stored in CBuffer0 it will be applied to ADC0 and if in CBuffer1 it applies to ADC1. If this indication is omitted the register applies for both ADC0 and ADC1, independent of the CBuffer used.
2. See the chip-specific EQADC information for details about which analog inputs have pull up/pull down control
3. Defined as an address for dummy write operation. Read is not allowed.

30.5.33.1 ADC0/1 Control Registers (ADC0_CR and ADC1_CR)

The ADC0/1 Control Registers (ADC0/1_CR) is used to define the standard configuration of the ADC. In the standard configuration, the parameters contained in the Alternate Configuration Control Registers (ADC_ACR1-14) are fixed at their reset value. A conversion uses the standard configuration when the conversion command (with the standard format) is written to address 0x00 of the on-chip ADC memory map. Refer to [Conversion command format for the standard configuration](#).

ADC0 Register address: 0x01

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	ADC0	0	0	0	ADC0	0	ADC0	ADC0	ADC0	ADC0	ADC0_CLK_PS					
W	_EN				_EMU		_TBSEL	_ODD	_CLK	_CLK						
				X			_PS	_PS	_DTY	_SEL						
RESET:	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1

ADC1 Register address: 0x01

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	ADC1	0	0	0	ADC1	0	ADC1	ADC1	ADC1	ADC1	ADC1_CLK_PS					
W	_EN				_EMU		_TBSEL	_ODD	_CLK	_CLK						
				X			_PS	_PS	_DTY	_SEL						
RESET:	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1

 = Unimplemented or Reserved

Figure 30-104. ADC0/1 Control Registers (ADC0/1_CR)

Table 30-112. ADC0/1_CR field descriptions

Field	Description
ADC0/1_EN	Enable bit for ADC0/1 ADC0/1_EN enables ADC0/1 to perform A/D conversions. See Enabling and disabling the on-chip ADCs for details. Note: The bias generator circuit inside the ADC hard macro ceases functioning when both ADC0_EN and ADC1_EN bits are negated. Note: Conversion commands sent to the CBuffer of a disabled ADC are ignored by the ADC control hardware. Note: When the ADC0/1_EN status is changed from asserted to negated, the ADC Clock will not stop until it reaches its low phase.
	0 ADC is disabled. Clock supply to ADC0/1 is stopped.
	1 ADC is enabled and ready to perform A/D conversions.
ADC0/1_EMUX	External Multiplexer enable for ADC0/1 When ADC0/1_EMUX is asserted, the MA pins will output digital values according to the number of the external channel being converted for selecting external multiplexer inputs. See Internal/external multiplexing for a detailed description about how ADC0/1_EMUX affects channel number decoding. Note: Both ADC0/1_EMUX bits must not be asserted at the same time. Note: The ADC0/1_EMUX bit must only be written when the ADC0/1_EN bit is negated. ADC0/1_EMUX can be set during the same write cycle used to set ADC0/1_EN.
	0 External multiplexer disabled; no external multiplexer channels can be selected.
	1 External multiplexer enabled; external multiplexer channels can be selected.

Table continues on the next page...

Table 30-112. ADC0/1_CR field descriptions (continued)

Field	Description
ADC0/1_TBSEL[0:1]	Timebase Selection for ADC0/1 The ADC0/1_TBSEL[0:1] field selects the time information to be used as timestamp according to Table 30-113. Note: This selection is overridden by the corresponding field ATBSEL in the ADC_ACR1-14 registers when the alternate conversion command is used.
ADC0/1_ODD_PS	Clock Prescaler Odd Rates Selector for ADC0/1 The ADC0/1_CLK_DTY field controls the duty rate of the ADC0/1 clock when the ADC0/1_CLK_PS field is asserted. The generated clock has an odd number of system clock cycles, therefore this field is used to select a clock duty higher or lower than 50%.
	0 Even divide factor is selected. The final divide factor is dependent of ADC0/1_CLK_PS field.
	1 Odd divide factor is selected. The final divide factor is dependent of ADC0/1_CLK_PS field.
ADC0/1_CLK_DTY	Clock Duty Rate Selector for ADC0/1 (for odd divide factors) The ADC0/1_ODD_PS field is used together with the ADC0/1_CLK_PS field to define even/odd divide factors in the generation of the ADC0/1 clocks. See Table 30-114 for available divide factors.
	0 clock low interval is longer 1 clock cycle than high pulse.
	1 clock high pulse is longer 1 clock cycle than low portion.
ADC0/1_CLK_SEL	Clock Selector for ADC0/1 The ADC0/1_CLK_SEL is used to select between the system clock signal or the prescaler output signal. The prescaler provides the system clock signal divided by a even factor from 2 to 64. This is required to permit the ADC to run as fast as possible when the device is in Low Power Active mode and system clock is around 1 MHz. Note: The ADC0/1_CLK_SEL bits must only be written when the ADC0/1_EN bit is negated. ADC0/1_CLK_SEL can be set during the same write cycle used to set ADC0/1_EN.
	0 Prescaler output clock is selected.
	1 System clock is selected - maximum frequency.
ADC0/1_CLK_PS[0:4]	Clock Prescaler Field for ADC0/1 The ADC0/1_CLK_PS field controls the system clock divide factor for the ADC0/1 clock as in Table 30-114. See ADC clock and conversion speed for details about how to set ADC0/1_CLK_PS. Note: The ADC0/1_CLK_PS field must only be written when the ADC0/1_EN bit is negated. This field can be configured during the same write cycle used to set ADC0/1_EN.

Table 30-113. Timebase selection

ADC0/1_TBSEL[0:1]	Definition
00	Selects internally generated time base as time stamp.
01	Selects imported time base 1 indicated by SRV1 bit field of EQADC_STACCCR register.
10	Selects imported time base 2 indicated by SRV2 bit field of EQADC_STACCCR register.

Table continues on the next page...

Table 30-113. Timebase selection (continued)

ADC0/1_TBSEL[0:1]	Definition
11	Reserved

Table 30-114. System Clock Divide Factor for ADC Clock

ADC0/1_CLK_PS[0:4]	System Clock Divide Factor	
	ADC0/1_ODD_PS = 0	ADC0/1_ODD_PS = 1
0b00000	2	3
0b00001	4	5
0b00010	6	7
0b00011	8	9
0b00100	10	11
0b00101	12	13
0b00110	14	15
0b00111	16	17
0b01000	18	19
0b01001	20	21
0b01010	22	23
0b01011	24	25
0b01100	26	27
0b01101	28	29
0b01110	30	31
0b01111	32	33
0b10000	34	35
0b10001	36	37
0b10010	38	39
0b10011	40	41
0b10100	42	43
0b10101	44	45
0b10110	46	47
0b10111	48	49
0b11000	50	51
0b11001	52	53
0b11010	54	55
0b11011	56	57
0b11100	58	59
0b11101	60	61
0b11110	62	63
0b11111	64	65

30.5.33.2 ADC Time Stamp Control Register (ADC_TSCR)

The ADC Time Stamp Control Register (ADC_TSCR) contains a system clock divide factor used in the making of the time base counter clock. It determines at what frequency the time base counter will run. ADC_TSCR can be accessed by configuration commands sent to CBuffer0 or to CBuffer1. A data write to ADC_TSCR through a configuration command sent to CBuffer0 will write the same memory location as when writing to it through a configuration command sent to CBuffer1.

Note

Simultaneous write accesses from CBuffer0 and CBuffer1 to ADC_TSCR are not allowed.

ADC0/1 Register address: 0x02

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	0	0	0	0	0	TBC_CLK_PS			
W																
RESET:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

 = Unimplemented or Reserved

Figure 30-105. ADC Time Stamp Control Register (ADC_TSCR)

Table 30-115. ADC_TSCR field descriptions

Field	Description
TBC_CLK_PS[0:3]	Time Base Counter Clock Prescaler The TBC_CLK_PS field contains the system clock divide factor for the time base counter. It controls the accuracy of the time stamp. The prescaler is disabled when TBC_CLK_PS is set to 0b0000. See Table 30-116. Note: If TBC_CLK_PS is not set to disabled, it must not be changed to any other value besides disabled. If TBC_CLK_PS is set to disabled it can be changed to any other value.

Table 30-116. Clock divide factor for time stamp

TBC_CLK_PS[0:3]	System clock divide factor	Clock to time stamp counter for a 120 MHz system clock (MHz)
0b0000	Disabled	Disabled
0b0001	1	120
0b0010	2	60
0b0011	4	30

Table continues on the next page...

Table 30-116. Clock divide factor for time stamp (continued)

TBC_CLK_PS[0:3]	System clock divide factor	Clock to time stamp counter for a 120 MHz system clock (MHz)
0b0100	6	20
0b0101	8	15
0b0110	10	12
0b0111	12	10
0b1000	16	7.5
0b1001	32	3.75
0b1010	64	1.88
0b1011	128	0.94
0b1100	256	0.47
0b1101	512	0.23
0b1110 - 0b1111	Reserved	-

30.5.33.3 ADC Time Base Counter Registers (ADC_TBCR)

The ADC Time Base Counter Register (ADC_TBCR) contains the current value of the time base counter. ADC_TBCR can be accessed by configuration commands sent to CBuffer0 or to CBuffer1. A data write to ADC_TBCR through a configuration command sent to CBuffer0 will write the same memory location as when writing to it through a configuration command sent to CBuffer1.

Note

Simultaneous write accesses from CBuffer0 and CBuffer1 to ADC_TBCR are not allowed.

ADC0/1 Register address: 0x03

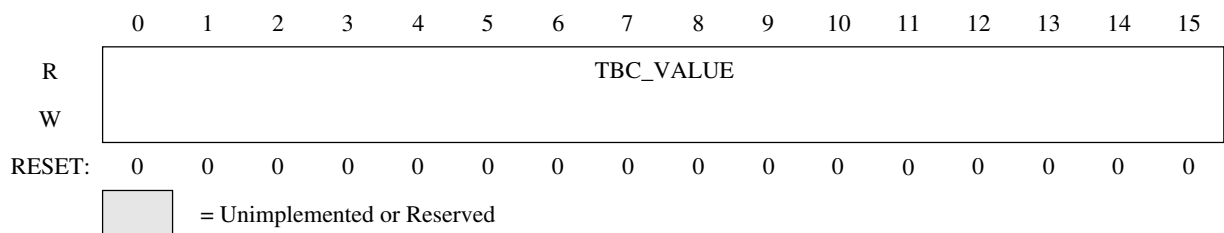
**Figure 30-106. ADC Time Base Counter Register (ADC_TBCR)**

Table 30-117. ADC_TBCR field descriptions

Field	Description
TBC_VALUE[0:15]	Time Base Counter VALUE Field The TBC_VALUE field contains the current value of the time base counter. Reading TBC_VALUE returns the current value of time base counter. Writes to TBC_VALUE register load the written data to the counter. The time base counter counts from 0x0000 to 0xFFFF and wraps when reaching 0xFFFF.

30.5.33.4 ADC0/1 Gain Calibration Constant Registers (ADC0_GCCR and ADC1_GCCR)

The ADC0/1 Gain Calibration Constant Register (ADC0/1_GCCR) contains the gain calibration constant used to fine-tune the ADC0/1 conversion results. See [ADC calibration feature](#) for details about the calibration scheme used in the EQADC.

ADC0 Register address: 0x04

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0															
W																
RESET:	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0

ADC1 Register address: 0x04

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0															
W																
RESET:	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0

 = Unimplemented or Reserved

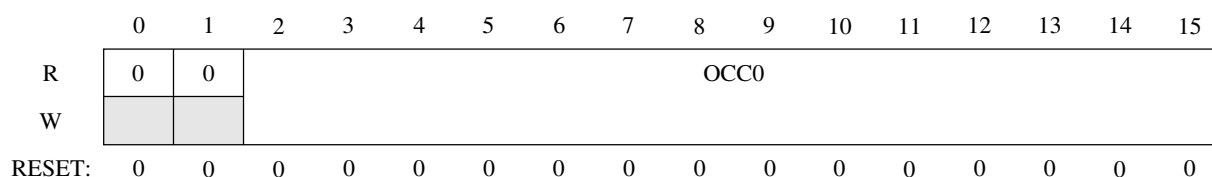
Figure 30-107. ADC0/1 Gain Calibration Constant Registers (ADC0/1_GCCR)**Table 30-118. ADC0/1_GCCR field descriptions**

Field	Description
GCC0/1[0:14]	Gain calibration constant for ADC0/1 GCC0/1 contains the gain calibration constant used to fine-tune ADC0/1 conversion results. It is a unsigned 15-bit fixed pointed value. The gain calibration constant is an unsigned fixed point number expressed in the GCC_INT.GCC_FRAC binary format. The integer part of the gain constant (GCC_INT) contains a single binary digit while its fractional part (GCC_FRAC) contains fourteen digits. For details about the GCC data format, see MAC unit and operand data format .

30.5.33.5 ADC0/1 Offset Calibration Constant Registers (ADC0_OCCR and ADC1_OCCR)

The ADC0/1 Offset Calibration Constant Register (ADC0/1_OCCR) contains the offset calibration constant used to fine-tune of ADC0/1 conversion results. The offset constant is a signed 14-bit integer value. See [ADC calibration feature](#) for details about the calibration scheme used in the EQADC.

ADC0 Register address: 0x05



ADC1 Register address: 0x05

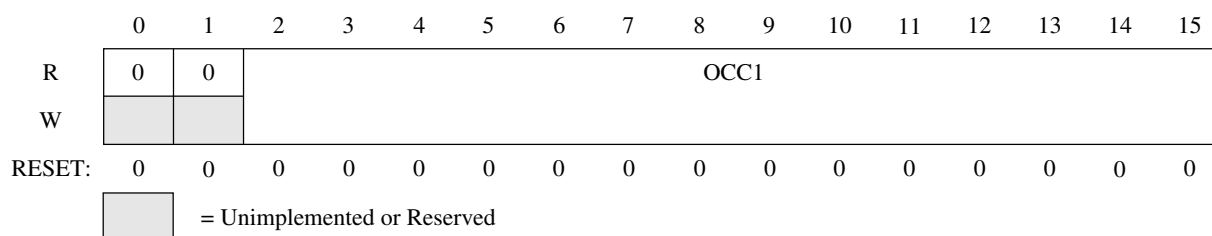


Figure 30-108. ADC0/1 Offset Calibration Constant Registers (ADC0/1_OCCR)

Table 30-119. ADC0/1_OCCR field descriptions

Field	Description
OCC0/1[0:13]	Offset Calibration Constant of ADC0/1 OCC0/1 contains the offset calibration constant used to fine-tune ADC0/1 conversion results. Negative values should be expressed using the two's complement representation.

30.5.33.6 Alternate Configuration 1-14 Control Registers (ADC_ACR1-14)

The Alternate Configuration Control Registers (ADC_ACR1-14) are used to configure the alternate configurations of the ADC. There are 14 possible alternate configurations, each one associated with one of the ADC_ACR1-14 registers. All alternate configurations share the same standard configuration parameters from the ADC0/1_CR

registers, plus additional configuration parameters contained in the ADC_ACR1-14. A conversion uses one of the alternate configurations when the conversion command (with the alternate configuration format) is written to an address in the range 0x08-0x15 of the on-chip ADC memory map. It also can use the Extended Alternated Configurations as described in [Extended Alternate Configuration 1-14 Control Registers \(ADC_EACR1-14\)](#) if bit DAM=1. See [Conversion command format for alternate configurations](#).

ADC0/1 Register address: 0x30
ADC0/1 Register address: 0x34
ADC0/1 Register address: 0x38
ADC0/1 Register address: 0x3C
ADC0/1 Register address: 0x40
ADC0/1 Register address: 0x44
ADC0/1 Register address: 0x48
ADC0/1 Register address: 0x4C
ADC0/1 Register address: 0x50
ADC0/1 Register address: 0x54
ADC0/1 Register address: 0x58
ADC0/1 Register address: 0x5C
ADC0/1 Register address: 0x60
ADC0/1 Register address: 0x64



¹ Reserved for expansion of DEST.

Figure 30-109. Alternate Configuration 1-14 Control Registers (ADC_ACR1-14)

Table 30-120. ADC_ACR1-14 field descriptions

Field	Description				
RET_INH	Result Transfer Inhibit / Decimation Filter Pre-Fill This bit is used to inhibit the transfer of the result data from the peripheral module to the result queue. When the module is a Decimation Filter, this bit sets the filter in a special mode (PRE-FILL) in which it does not generate decimated samples out from the conversion results received from the EQADC block, but the conversion samples are used by the filter algorithm. This feature allows a proper initialization of the Decimation Filter without generating any decimated result. Or this bit is useful for sending the result of the ADC to the STAC bus master but not putting the result in the result queue. Note: Refer to the block guide of each peripheral module to obtain the correct behavior of the companion module to the PRE-FILL mode. <table><tr><td>0</td><td>Result transfer to result queue / Decimation Filter in filtering mode</td></tr><tr><td>1</td><td>No result transfer to result queue / Decimation Filter PRE-FILL mode</td></tr></table>	0	Result transfer to result queue / Decimation Filter in filtering mode	1	No result transfer to result queue / Decimation Filter PRE-FILL mode
0	Result transfer to result queue / Decimation Filter in filtering mode				
1	No result transfer to result queue / Decimation Filter PRE-FILL mode				

Table continues on the next page...

Table 30-120. ADC_ACR1-14 field descriptions (continued)

Field	Description
DEST[0:3]	Conversion Result Destination Selection The DEST[0:3] field selects the primary destination of the conversion result generated by the Alternate Conversion Command as shown in Table 30-121. This field also affects the behavior of the FMTA bit and the FFMT bit of the conversion command for alternate configurations (see Conversion command format for alternate configurations). And this field is affected by DAM bit of the EQADC Module Configuration Register (see EQADC Module Configuration Register (EQADC_MCR)). There is also configurations to send the conversion result to a second destination using the Extended Alternate Configuration registers (Extended Alternate Configuration 1-14 Control Registers (ADC_EACR1-14)).
FMTA	Conversion Data Format for Alternate Configuration If the DEST field is not 0b000, the FMTA bit specifies how the 12-bit conversion data returned by the ADCs is formatted into the 16-bit data which is sent to the parallel side interface.
	0 Right justified unsigned
	1 Right justified signed
RPSI	Return Parallel Side Interface The RPSI bit is used in the companion modules that have two PSI interfaces (PSI0 and PSI1), which are connected to two eQADCs. It specifies in the companion module if the data received from one PSI should be transmitted to the same PSI or to the other one. Note: The companion module may take samples from two eQADCs, allowing double sample rate. If the RPSI bits are different, the results are sent to just one of the eQADCs. Note: It is not allowed to use RPSI=1 when the SoC has only one eQADC. It can occur errors in the companion module due to routing error.
	0 Companion module results from this eQADC are sent back to the same eQADC using the same PSI.
	1 Companion module results from this eQADC are sent to the other PSI
RESSEL[0:1]	ADC Resolution Selection The RESSEL[0:1] field selects the resolution of the ADC.
	00 ADC set to 12-bits resolution
	01 ADC set to 10-bits resolution
	10 ADC set to 8-bits resolution
	11 Reserved
ATBSEL[0:1]	Alternate Command Timebase Selector The ATBSEL[0:1] field selects the time information to be used as timestamp Note: This selection overrides the corresponding fields ADC0/1_TBSEL in the ADC0/1_CR registers when the alternate conversion command is used.
	00 Selects internally generated time base as time stamp
	01 Selects imported time base 1 indicated by SRV1 bit field of EQADC_STACCCR register.
	10 Selects imported time base 2 indicated by SRV2 bit field of EQADC_STACCCR register.
	11 Reserved

Table continues on the next page...

Table 30-120. ADC_ACR1-14 field descriptions (continued)

Field	Description
PRE_GAIN[0:1]	ADC Pre-gain control The PRE_GAIN[0:1] controls the gain of the ADC input stage by changing the internal ADC iterations in the gain stage.
	00 X1 gain
	01 X2 gain
	10 X4 gain
	11 Reserved

Table 30-121. Conversion destination selection

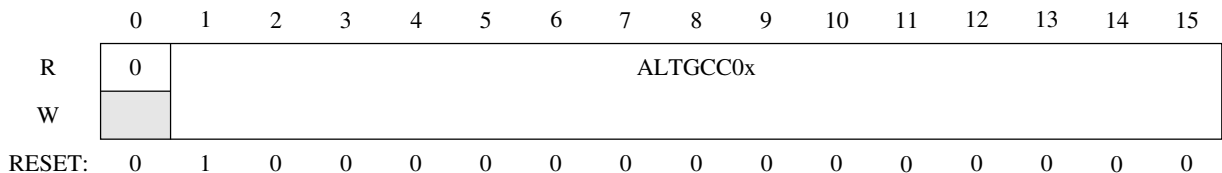
DAM	DEST[0:3]	First Destination	Second Destination	Description
0	0000	RFIFOn	-	The conversion result is sent to the RFIFOn (where n = MESSAGE_TAG). The data format is specified by the FFMT bit in the Conversion Command Message.
	0001 - 0111	1-7	-	The conversion result is sent to the Parallel Side Interface. Up to seven devices can be connected to the parallel interface, using the DEST field to select which one receives the conversion result. The data format is specified by the FMTA bit in the Alternate Configuration Control Register.
	1000	1	5	The conversion result is sent to the Parallel Side Interface to two destinations. The transferred data is the same for both destinations. - The data format is specified by the FMTA bit in the Alternate Configuration Control Register. - The FLUSH command is sent to both destinations.
	1001	2		
	1010	3		
	1011	4		
	1100	1	7	
	1101	2		
	1110	3		
	1111	4		
1	0000	RFIFOn	Depends on DEST2 (Table 30-125)	For DEST = 0, the conversion result is sent to the RFIFOs. The data format is specified by the FFMT bit in the Conversion Command Message. Note: It is not possible to have FLUSH control for DEST2 >0 because FFMT is used for format of DEST, not for FLUSH.
	0001 - 1111	1-15		The conversion result is sent to the Parallel Side Interface. Up to fifteen devices can be connected to the parallel interface, using the DEST field to select which one receives the conversion result. The data format is specified by the FMTA bit in the Alternate Configuration Control Register.

30.5.33.7 ADC0/1 Alternate Gain Registers (ADC0_AGR1-2 and ADC1_AGR1-2)

The Alternate Gain Registers (ADC0_AGRx and ADC1_AGRx, x=1-2) contain the gain calibration constants used to fine-tune the ADCs conversion results for alternate configurations 1 or 2. A conversion from an ADC uses the corresponding ADC0_AGRx or ADC1_AGRx register when the conversion command (with the alternate configuration format) is written to an address in the range 0x08-0x09 of the on-chip ADC memory map. See [ADC calibration feature](#) for details about the calibration scheme used in the EQADC.

ADC0 Register address: 0x31

ADC0 Register address: 0x35



ADC1 Register address: 0x31

ADC1 Register address: 0x35

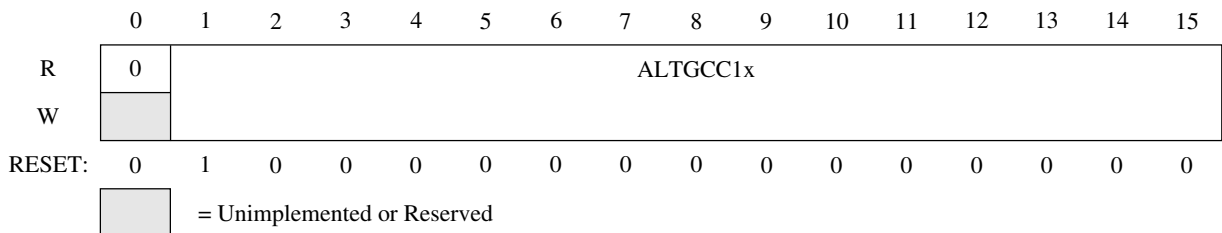


Figure 30-110. ADC0/1 Alternate x Gain Register (ADC0/1_AGRx, x=1-2)

Table 30-122. ADC0/1_AGRx field descriptions

Field	Description
ALTGCC0/1x[0:14]	Alternate Gain Calibration Constant ALTGCC0/1x[0:14] contain the gain calibration constants used to fine-tune ADC0/1 conversion results for alternate configurations 1 and 2. The gain calibration constants are 15-bit unsigned fixed point numbers expressed in the <i>GCC_INT.GCC_FRAC</i> binary format. The integer part of the gain constants (GCC_INT) contain a single binary digit while their fractional part (GCC_FRAC) contain fourteen digits. For details about the GCC data format, see MAC unit and operand data format.

30.5.33.8 ADC0/1 Alternate Offset Register (ADC0_AOR1-2 and ADC1_AOR1-2)

The Alternate Offset Registers (ADC0_AORx and ADC1_AORx, x=1-2) contain the offset calibration constants used to fine-tune ADCs conversion results for alternate configurations 1 and 2. The offset constants are signed 14-bit integer values. See [ADC calibration feature](#) for details about the calibration scheme used in the EQADC.

ADC0 Register address: 0x32
ADC0 Register address: 0x36

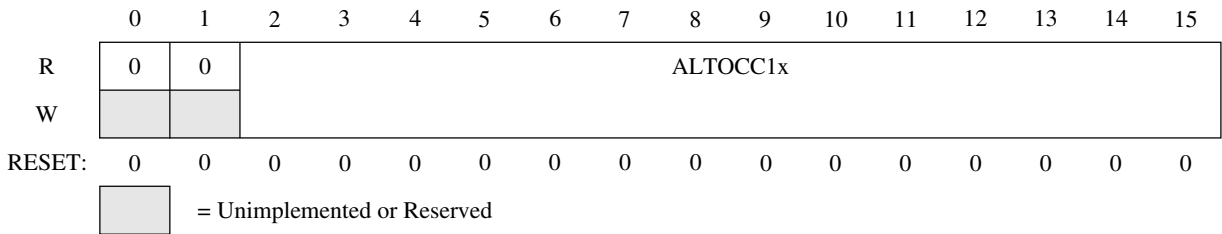
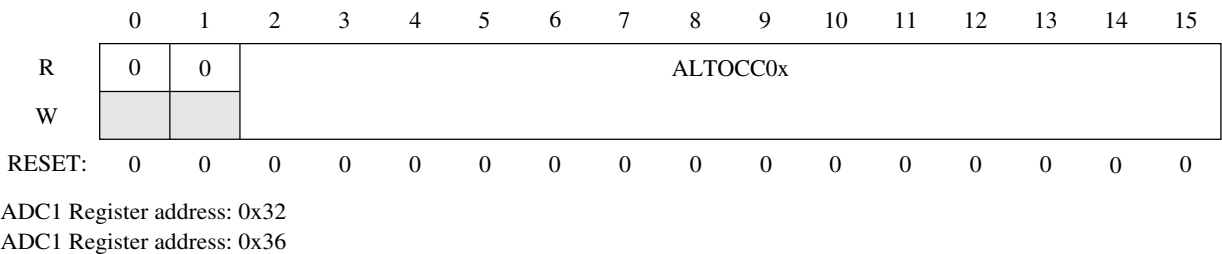


Figure 30-111. ADC0/1 Alternate x Offset Registers (ADC0/1_AORx, x=1-2)

Table 30-123. ADC0/1_AORx field descriptions

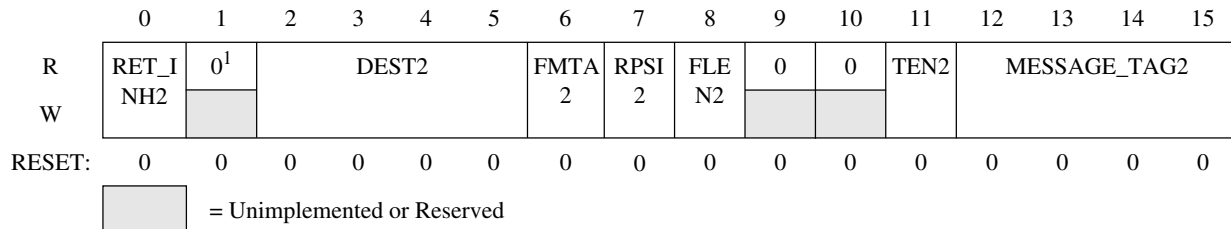
Field	Description
ALTOCC0/1x[0:13]	Alternate Offset Calibration Constant ALTOCC0/1x[0:13] contain the offset calibration constants used to fine-tune ADCs conversion results for alternate configurations 1 or 2. Negative values should be expressed using the two's complement representation.

30.5.33.9 Extended Alternate Configuration 1-14 Control Registers (ADC_EACR1-14)

The Extended Alternate Configuration Control Registers (ADC_EACR1-14) are used together with the Alternate Configuration Control Registers (ADC_ACR1-14) of the same number to complement the alternate configurations of the ADC. A conversion uses one pair of the alternate configuration regs (normal and extended) when the conversion

command (with the alternate conversion format) is written to an address in the range 0x08-0x15 of the on-chip ADC memory map. See [Conversion command format for alternate configurations](#).

ADC0/1 Register address: 0x33
 ADC0/1 Register address: 0x37
 ADC0/1 Register address: 0x3B
 ADC0/1 Register address: 0x3F
 ADC0/1 Register address: 0x43
 ADC0/1 Register address: 0x47
 ADC0/1 Register address: 0x4B
 ADC0/1 Register address: 0x4F
 ADC0/1 Register address: 0x53
 ADC0/1 Register address: 0x57
 ADC0/1 Register address: 0x5B
 ADC0/1 Register address: 0x5F
 ADC0/1 Register address: 0x63
 ADC0/1 Register address: 0x67



¹ Reserved for expansion of DEST2.

Figure 30-112. Extended Alternate Configuration 1-14 Control Registers (ADC_EACR1-14)

Table 30-124. ADC_EACR1-14 field descriptions

Field	Description
RET_INH2	Result Transfer Inhibit 2 / Decimation Filter Pre-Fill 2 Same definition of DEST field but to the peripheral module indicated by DEST2.
	0 Result transfer to result queue / Decimation Filter in filtering mode
	1 No result transfer to result queue / Decimation Filter PRE-FILL mode
DEST2[0:3]	Conversion Result Destination 2 Selection The DEST2[0:3] field selects the second destination of the conversion result generated by the Alternate Conversion Command. However this selection only can occur when DAM = 1 (see EQADC Module Configuration Register (EQADC_MCR)), as shown in previous Table 30-121 for DEST field. In addition, the eQADC operation due to this field is also affected by the ADC_ACRx DEST field, by the ADC_EACRx TEN2 field, by MESSAGE_TAG field, by the ADC_EACRx MESSAGE_TAG2 field as defined in Table 30-125 .
FMTA2	Conversion Data Format for Alternate Configuration DEST2 The FMTA2 bit specifies how the 12-bit conversion data returned by the ADCs is formatted into the 16-bit data which is sent to the RFIFOs or the parallel side interface defined by DEST2. See Table 30-125 for more details.
	0 Right justified unsigned

Table continues on the next page...

Table 30-124. ADC_EACR1-14 field descriptions (continued)

Field	Description
	1 Right justified signed
RPSI2	Return Parallel Side Interface for DEST2 The RPSI2 bit is used in the companion module that has two PSI interfaces (PSI0 and PSI1) that are connected to two eQADCs. It specifies in the companion module if the data received from one PSI should be transmitted to the same PSI or to the other one. Note: The companion module may take samples from two eQADCs, allowing double sample rate. If the RPSI bits are different, the results are sent to just one of the eQADCs. Note: It is not allowed to use RPSI2=1 when the SoC has only one eQADC. It can occur errors in the companion module due to routing error.
	0 Companion module results from this eQADC are sent back to the same eQADC using the same PSI.
	1 Companion module results from this eQADC are sent to the other PSI
FLEN2	Flush Enable to DEST2 The FLEN2 bit enables the transfer of the flush command given by the FFMT field from the alternate conversion command when DEST>0.
	0 No flush command is transmitted to DEST2.
	1 Flush command is sent to DEST2 if specified by FFMT when DEST>0.
TEN2	Enable MESSAGE_TAG2 to DEST2 This bit specifies if the MESSAGE_TAG2 value should be used to the result transfer of DEST2. See Table 30-125 for more details.
	0 MESSAGE_TAG2 is disabled.
	1 MESSAGE_TAG2 is enabled.
MESSAGE_TAG2[0:3]	MESSAGE TAG 2 Field Same as the MESSAGE_TAG field in the Conversion Command as described in Conversion command format for the standard configuration. Table 30-129 describes the meaning of the MESSAGE_TAG2.

Table 30-125. Conversion Destinations Selection (DAM = 1)

DEST [0:3]	DEST2 [0:3]	TEN 2	MESSAGE _TAG2 = MESSAGE _TAG	(FMTA2=FFMT) or (FMTA2=FMTA) & (RET2=RET) & (RPSI2=RPSI) & (Flush2=Flush)	First Dest.	Second Dest.	Description
0000	0000	0	X	X	RFIFOn	-	The conversion result is sent to the RFIFOn (where n = MESSAGE_TAG), and the data format is specified by the FFMT bit in the alternate conversion command.
		1	1	1			
				0		RFIFOn	- Same as above but the conversion result is sent again to RFIFOn if FMTA2 is different from FFMT. - OBS: Two values are sent to the same RFIFO for one ADC conversion.
			0	X		RFIFOy	- The conversion result is sent to the RFIFOn (where n = MESSAGE_TAG), and the data format is specified by the FFMT bit in the alternate conversion command. - The conversion result is also sent to RFIFOy (where y = MESSAGE_TAG2) and the data format is specified by the FMTA2.

Table continues on the next page...

Table 30-125. Conversion Destinations Selection (DAM = 1) (continued)

DEST [0:3]	DEST2 [0:3]	TEN 2	MESSAGE _TAG2 = MESSAGE _TAG	(FMTA2=FFMT) or (FMTA2=FMTA) & (RET2=RET) & (RPSI2=RPSI) & (Flush2=Flush)	First Dest.	Second Dest.	Description
0000	0001 - 1111	0	X	X	RFIFOn	1 - 15	- The conversion result is sent to the RFIFOn (where n = MESSAGE_TAG), and the data format is specified by the FFMT bit. - The conversion result is also sent to PSI using the DEST2 field and the MESSAGE_TAG value. The data format is specified by the FMTA2 bit. There is not flush command. RET2 and RPSI2 are also used. - OBS: If the companion block addressed by DEST2 sends data back to the same eQADC, this result is sent to RFIFOn (two values to the same RFIFO for one ADC conversion). Note: It is not possible to have FLUSH control for DEST2>0 because FFMT is used for format of DEST, not for FLUSH.
		1	1	X			
			0	X			Same as above, but if the companion block addressed by DEST2 sends data back to the eQADC, this data is sent to RFIFOy given by MESSAGE_TAG2.

Table continues on the next page...

Table 30-125. Conversion Destinations Selection (DAM = 1) (continued)

DEST [0:3]	DEST2 [0:3]	TEN 2	MESSAGE _TAG2 = MESSAGE _TAG	(FMTA2=FFMT) or (FMTA2=FMTA) & (RET2=RET) & (RPSI2=RPSI) & (Flush2=Flush)	First Dest.	Second Dest.	Description
0001 - 1111	0000	0	X	X	1 - 15	-	The conversion result is sent to PSI using the DEST field and the MESSAGE_TAG value. The data format is specified by the FMTA bit. The flush command depends on FFMT command field.
		1	1	X		RFIFOn	- The conversion result is sent to PSI using the DEST field and the MESSAGE_TAG value. The data format is specified by the FMTA bit. The flush command depends on FFMT command field. - The conversion result is also sent to the RFIFOn (where n = MESSAGE_TAG2), and the data format is specified by the FMTA2 bit field. - OBS: If the companion block addressed by DEST sends data back to the same eQADC, this result is sent to RFIFOn (two values to the same RFIFO for one ADC conversion).
			0	X		RFIFOy	Same as above, but if the companion block addressed by DEST sends data back to the eQADC, this data is sent to RFIFOn=MESSAGE_TAG.

Table continues on the next page...

Table 30-125. Conversion Destinations Selection (DAM = 1) (continued)

DEST [0:3]	DEST2 [0:3]	TEN 2	MESSAGE _TAG2 = MESSAGE _TAG	(FMTA2=FFMT) or (FMTA2=FMTA) & (RET2=RET) & (RPSI2=RPSI) & (Flush2=Flush)	First Dest.	Second Dest.	Description
0001 - 1111	0001 - 1111 (Equal to DEST)	0	X	1	1 - 15	-	The conversion result is sent to PSI using the DEST field and the MESSAGE_TAG value. The data format is specified by the FMTA bit. The flush command depends on FFMT command field.
				0			
		1	1	0		1 - 15 (Equal to 1st Dest)	- Same as above. - The conversion result is sent again to PSI to the same DEST=DEST2. However, the PSI data bus value is different. - OBS: The data related to DEST is sent first. And the data related to DEST2 is sent later.
				1			
						-	The conversion result is sent to PSI using the DEST field and the MESSAGE_TAG value. The data format is specified by the FMTA bit. The flush command depends on FFMT command field.
				0		1 - 15 (Equal to 1st Dest)	- Same as above. - The conversion result is also sent to PSI using the DEST2=DEST field and with the MESSAGE_TAG2 value. The data format is specified by the FMTA2 bit. The flush command depends on FLEN2 bit field. RET2 and RPSI2 are also used.

Table continues on the next page...

Table 30-125. Conversion Destinations Selection (DAM = 1) (continued)

DEST [0:3]	DEST2 [0:3]	TEN 2	MESSAGE _TAG2 = MESSAGE _TAG	(FMTA2=FFMT) or (FMTA2=FMTA) & (RET2=RET) & (RPSI2=RPSI) & (Flush2=Flush)	First Dest.	Second Dest.	Description
0001 - 1111	0001 - 1111 (Not equal to DEST)	0	X	0	1-15	1 - 15 (Not equal to 1st Dest)	- The conversion result is sent to PSI using the DEST field and the MESSAGE_TAG value. The data format is specified by the FMTA bit. The flush command depends on FFMT command field. - The conversion result is also sent to PSI using the DEST2 (not equal DEST) field and with the MESSAGE_TAG2 value. The data format is specified by the FMTA2 bit. The flush command depends on FLEN2 bit field. RET2 and RPSI2 are also used. - OBS1: If the companion blocks addressed by DEST and DEST2 send data back to the same eQADC, these results are sent to the same RFIFO (two values to the same RFIFO for one ADC conversion). - OBS2: The data related to DEST is sent first. And the data related to DEST2 is sent later. Note: When sending timestamp in this case, the values sent to RFIFO can be out of order due to the prioritization of reading in the receiving part of the PSI sub-block.
				1			Same as above, except OBS2. In this case, the data is sent to DEST and DEST2 at the same time.
		1	1	1			Same as first case above.
				0			
			0	X			Same as first case, except OBS1 and Note2.

30.5.33.10 ADC Pull Up/Down Control Register x (ADC_PUDCRx, x=0-39)

The ADC Pull Up/Down Control Register x (ADC_PUDCR x) contains configuration bits for pull up and pull down resistors present at ADC input channels x , $x=0$ to 39.

ADC0/1 Register address: 0x70-0x97

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	CH_PULL x		0	0	PULL_STR x		0	0	0	0	0	0	0	0
W																
RESET:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

 = Unimplemented or Reserved

Figure 30-113. ADC Pull Up/Down Control Register x (ADC_PUDCR x , $x=0-39$)

Table 30-126. ADC_PUDCR x field descriptions

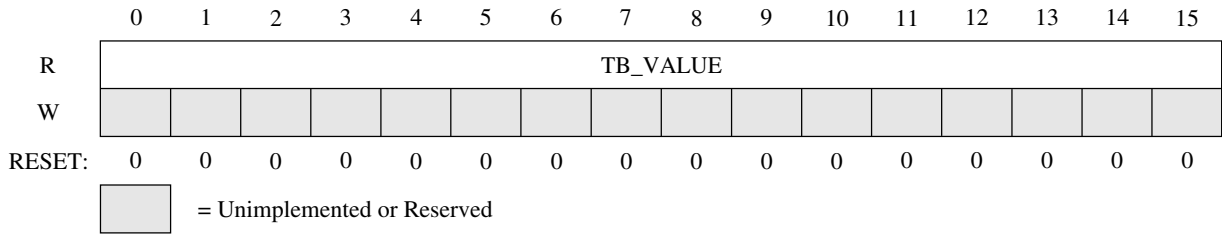
Field	Description
CH_PULL x [0:1]	Channel x Pull Up/Down Control bits The CH_PULL x [0:1] field controls the pull up/down configuration of the channel x .
	00 No Pull resistors connected to the channel
	01 Pull Up resistor connected to the channel
	10 Pull Down resistor connected to the channel
	11 Pull Up and Pull Down resistors connected to the channel
PULL_STR x [0:1]	Pull Up/Down Strength Control bits of channel x The PULL_STR x [0:1] bit field defines the strength of the channel x pull up or down resistors.
	00 Reserved
	01 200 Kohms pull resistor
	10 100 Kohms pull resistor
	11 5 Kohms (Approx.) pull resistor (this set is not available for CH_PULL $_x$ = 11)

30.5.33.11 STAC Bus Time Base 1/2 Registers (ADC_STACTBR1/2)

The STAC Bus Time Base Register1/2 (ADC_STACTBR1/2) contain the current value of the imported time base counter as selected by bit fields on register

EQADC_STACCCR. See [Conversion command format for the standard configuration](#) for more details.

ADC0/1 Register address: 0xA0, 0xA1

**Figure 30-114. ADC Time Base Counter Register (ADC_TBCR)****Table 30-127. ADC_TBCR field descriptions**

Field	Description
TB_VALUE[0:15]	Time Base VALUE Field The TB_VALUE field contains the current value of the imported time base.

30.6 Functional description

30.6.1 Overview

The EQADC provides a parallel interface to two on-chip ADCs and a parallel side interface to on-chip companion modules, like a decimation filter. The two on-chip ADCs are architected to allow access to all the analog channels.

Initially, command data is contained in system memory in a user defined data structure which is likely to be a queue as depicted in [Figure 30-1³](#). Command data is moved between the CQueues and CFIFOs by the host CPU or by the DMAC which respond to interrupt and DMA requests generated by the EQADC. The EQADC supports software and hardware triggers from other blocks or external pins to initiate transfers of commands from the multiple CFIFOs to the on-chip ADCs.

CFIFOs can be configured in single-scan or continuous-scan mode. When a CFIFO is configured in single-scan mode, the EQADC scans the CQueue one time. The EQADC stops transferring commands from the triggered CFIFO after detecting the EOQ bit set in the last transfer. After an EOQ bit is detected, software involvement is required to rearm the CFIFO so that it can detect new trigger events.

3. Command and result data can be stored in system memory in any user defined data structure. However, in this document it will be assumed that the data structure of choice is a queue, since it is the most likely data structure to be used and because queues are the only type of data structure supported by the DMAC.

When a CFIFO is configured for continuous-scan mode, the whole CQueue is scanned multiple times. After the detection of an asserted EOQ bit in the last command transfer, command transfers can continue or not depending on the mode of operation of the CFIFO.

CFIFO0 has a special configuration option to allow a repetitive sequence of conversion commands (streaming mode) with high priority characteristics (abort operation) or not. This feature is useful with the immediate conversion command feature that allows the immediate execution of a conversion command or a sequence of commands with critical timing even with the possibility of abortion of some current ADC conversion in progress. The aborted command is stored and executed again as soon as the critical timing commands have been finished.

The multiple Result FIFOs (RFIFOs) can receive data from the on-chip ADCs or from on-chip companion modules. Data from the on-chip ADCs can be routed to the side interface, processed by the on-chip companion modules and then routed back through the side interface to the RFIFOs.

30.6.2 Data flow in EQADC

30.6.2.1 Overview and basic terminology

Figure 30-115 shows how command data flows inside the EQADC system. A Command Message is the predefined format at which command data is stored on the CQueues. A Command message has 32 bits and is composed of two parts: a CFIFO header and an ADC Command. Command messages are moved from the CQueues to the CFIFOs by the host CPU or by the DMAC as they respond to interrupt and DMA requests generated by the EQADC. The EQADC generates these requests whenever a CFIFO is not full. *The FIFO Control Unit* will only transfer to a CBuffer the ADC command part of the Command Message. Information in the CFIFO header together with the upper bit of the ADC command is used by the *FIFO Control Unit* to arbitrate which triggered CFIFO will be transferring the next command.

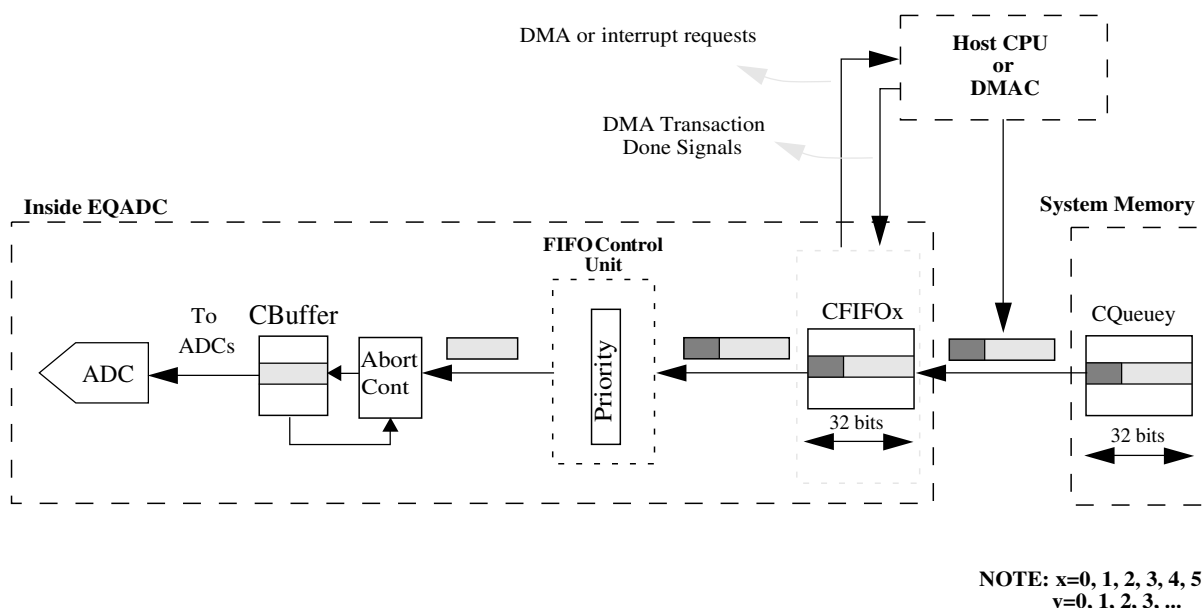


Figure 30-115. Command Flow during EQADC operation

ADC commands sent to the on-chip CBuffers are executed in a first-in-first-out basis with exception when the immediate conversion command function is enabled. Three types of results can be expected: data read from an ADC register, a conversion result, or a time stamp.

Note

While the EQADC pops commands out from a CFIFO, it also is checking the number of entries in the CFIFO and generating requests to fill it. The process of pushing and popping commands to and from a CFIFO can occur simultaneously. However, this is not true for CFIFO0 when configured to operate in streaming mode for popping.

The *FIFO Control Unit* expects all incoming results to be shaped in a predefined Result Message format. [Figure 30-116](#) shows how result data flows inside the EQADC system. Results generated on the on-chip ADCs are adjusted considering the selected resolution of the ADC and are formatted into result messages inside the *Result Format and Calibration Sub-Block*. This result message can be routed directly to one of the RFIFOs or to an on-chip companion module via the parallel side interface. After the data is processed by the companion module, it can be routed back to one of the RFIFOs via the side interface with the correct format. A result message is composed of an RFIFO header

and an ADC Result. The *FIFO Control Unit* decodes the information contained in the RFIFO header to determine the RFIFO to which the ADC result should be sent. Once in an RFIFO, the ADC result is moved to the corresponding RQueue by the host CPU or by the DMAC as they respond to interrupt and DMA requests generated by the EQADC. The EQADC generates these requests whenever an RFIFO has at least one entry.

Note

While conversion results are returned, the EQADC is checking the number of entries in the RFIFO and generating requests to empty it. The process of pushing and popping ADC results to and from an RFIFO can occur simultaneously.

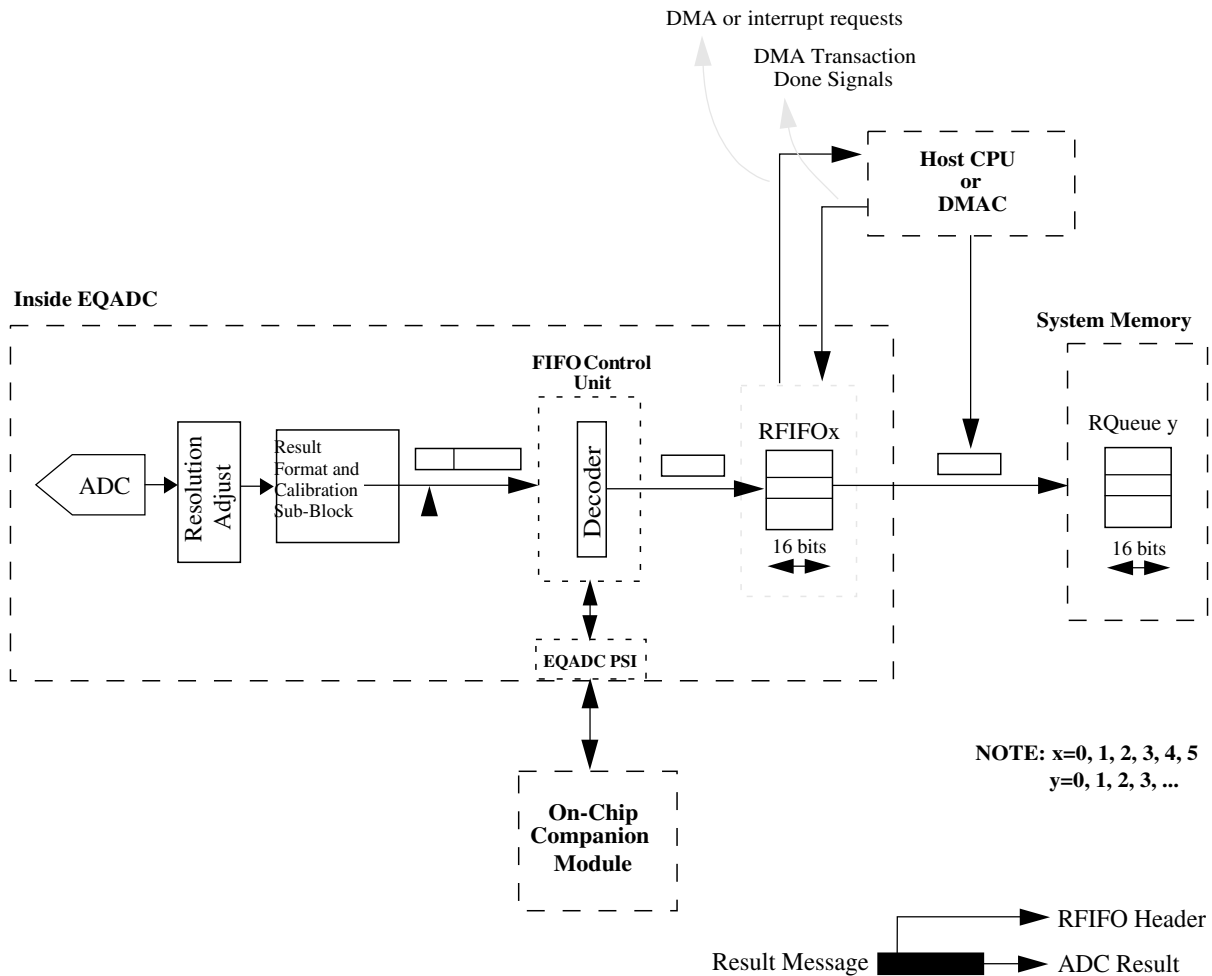


Figure 30-116. Result Flow during EQADC operation

30.6.2.2 Message format in EQADC

This section explains the command and result message formats used for on-chip ADC operation

A Command Message is the predefined format at which command data is stored on the CQueues. A Command message has 32 bits and is composed of two parts: a CFIFO header and an ADC Command. The size of the CFIFO header is fixed to 6 bits, and it works as inputs to the *FIFO Control Unit*. It controls when a CQueue ends, when it pauses. Information contained in the CFIFO header, together with the upper bit of the ADC Command is used by the *FIFO Control Unit* to arbitrate which triggered CFIFO will transfer the next command. ADC commands are encoded inside the least significant 26 bits of the command message.

A Result message is composed of an RFIFO header and an ADC Result. The *FIFO Control Unit* decodes the information contained in the RFIFO header to determine the RFIFO to which the ADC result should be sent. An ADC result is always 16 bits long.

30.6.2.2.1 Message Formats for On-Chip ADC Operation

This section describes the Command/Result message formats used for on-chip ADC operation.

30.6.2.2.1.1 Conversion command format for the standard configuration

Figure 30-117 describes the format for conversion commands when interfacing with the on-chip ADCs in the standard configuration. The standard configuration is selected when the lowest byte (bits 24-31) of the conversion command is set to zero. In the standard configuration, the conversion result is always routed to one of the RFIFOs. A time stamp information can be optionally requested.

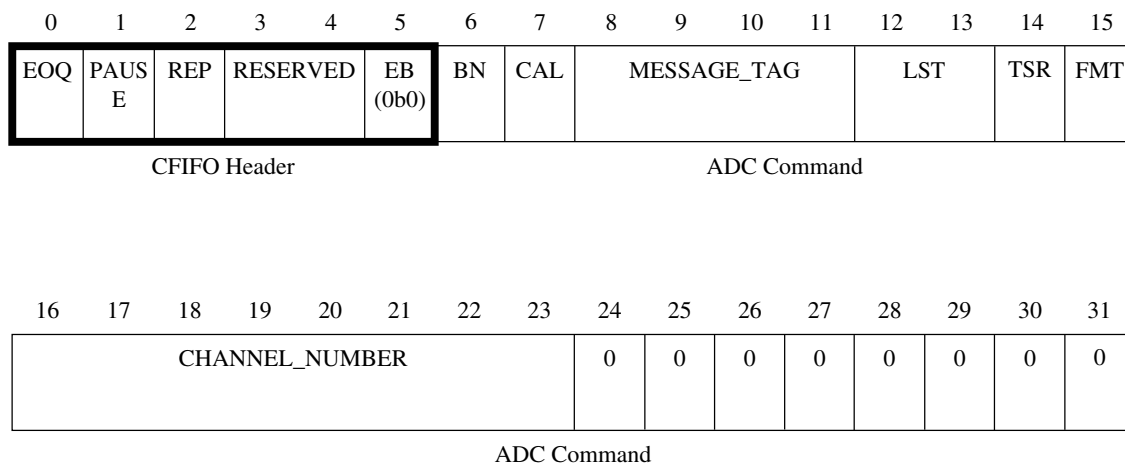


Figure 30-117. Conversion command format for the standard configuration

**Table 30-128. Conversion command format for the standard configuration
— field descriptions**

Field	Description
EOQ	End Of Queue Bit NOTE: The EOQ bit is asserted in the last command of a CQueue to indicate to the EQADC that a scan of the CQueue is completed. EOQ instructs the EQADC to reset its current CFIFO transfer counter value (TC_CF) to zero. Depending on the CFIFO mode of operation, the CFIFO status will also change upon the detection of an asserted EOQ bit on the last transferred command - see CFIFO scan trigger modes for details. If both the PAUSE and EOQ bits are asserted in the same command message the respective flags are set, but the CFIFO status changes as if only the EOQ bit were asserted.
	0 Not the last entry of the CQueue.
	1 Last entry of the CQueue.
PAUSE	Pause Bit The Pause bit allows software to create sub-queues within a CQueue. When the EQADC completes the transfer of a command with an asserted Pause bit, the CFIFO enters the WAITING FOR TRIGGER state. See CFIFO operation status for a description of the state transitions. The Pause bit is only valid when CFIFO operation mode is configured to single or continuous-scan edge trigger mode. Note: If both the PAUSE and EOQ bits are asserted in the same command message the respective flags are set, but the CFIFO status changes as if only the EOQ bit were asserted.
	0 Do not enter WAITING FOR TRIGGER state after transfer of the current Command Message.
	1 Enter WAITING FOR TRIGGER state after transfer of the current Command Message.
REP	Repeat/loop Start Point Indication Bit The REP bit is asserted in the command to indicate where is the start point of the sub-queue to be repeated when the streaming mode is enabled. The PAUSE bit indicates the end point of the sub-queue. Therefore, both can occur in the same command or in separated ones. If two or more REP bits are read before a PAUSE bit, this is an error case and the intermediary REP bits are ignored.
	0 It is not the start point of a loop.
	1 Indicates the start point of the sub-queue to be repeated.
Reserved	Reserved
BN	Buffer Number Bit BN indicates which buffer the message will be stored in.
	0 Message stored in buffer 0.
	1 Message stored in buffer 1.
CAL	CALibration Bit CAL indicates if the returning conversion result must be calibrated.
	0 Do not calibrate conversion result.
	1 Calibrate conversion result.

Table continues on the next page...

Table 30-128. Conversion command format for the standard configuration — field descriptions (continued)

Field	Description								
MESSAGE_TAG[0:3]	MESSAGE_TAG Field The MESSAGE_TAG allows the EQADC to separate returning results into different RFIFOs. Table 30-129 describes the meaning of the MESSAGE_TAG. When the EQADC transfers a command, the MESSAGE_TAG is included as part of the command. Eventually on-chip ADC returns the result with the same MESSAGE_TAG. The EQADC separates incoming messages into different RFIFOs by decoding the MESSAGE_TAG of the incoming data. Note: The MESSAGE_TAG value can be particularly used in a companion module (that uses the PSI) as the address to the annexed conversion result. The data can be routed to different sub-blocks based on the MESSAGE_TAG, similar to what is done with the RFIFOs in the eQADC. Additionally, in the companion module it is not required to follow Table 30-129.								
LST[0:1]	Long Sampling Time These two bits determine the duration of the sampling time in ADC clock cycles. <table> <tr> <td>00</td><td>2 sampling cycles (ADC clock cycles)</td></tr> <tr> <td>01</td><td>8 sampling cycles (ADC clock cycles)</td></tr> <tr> <td>10</td><td>64 sampling cycles (ADC clock cycles)</td></tr> <tr> <td>11</td><td>128 sampling cycles (ADC clock cycles)</td></tr> </table>	00	2 sampling cycles (ADC clock cycles)	01	8 sampling cycles (ADC clock cycles)	10	64 sampling cycles (ADC clock cycles)	11	128 sampling cycles (ADC clock cycles)
00	2 sampling cycles (ADC clock cycles)								
01	8 sampling cycles (ADC clock cycles)								
10	64 sampling cycles (ADC clock cycles)								
11	128 sampling cycles (ADC clock cycles)								
TSR	Time Stamp Request TSR indicates the request for a time stamp. When TSR is asserted, the on-chip <i>ADC Control Logic</i> returns a time stamp for the current conversion command after the conversion result is sent to the RFIFOs. See Time stamp feature for details. <table> <tr> <td>0</td><td>Return conversion result only.</td></tr> <tr> <td>1</td><td>Return conversion time stamp after the conversion result.</td></tr> </table>	0	Return conversion result only.	1	Return conversion time stamp after the conversion result.				
0	Return conversion result only.								
1	Return conversion time stamp after the conversion result.								
FMT	Conversion Data Format FMT specifies to the EQADC how to format the 12-bit conversion data returned by the ADCs into the 16-bit format which is sent to the RFIFOs. See ADC result format for on-chip ADC operation for details. <table> <tr> <td>0</td><td>Right justified unsigned.</td></tr> <tr> <td>1</td><td>Right justified signed.</td></tr> </table>	0	Right justified unsigned.	1	Right justified signed.				
0	Right justified unsigned.								
1	Right justified signed.								
CHANNEL_NUMBER[0:7]	Channel Number Field The CHANNEL_NUMBER field selects the analog input channel. The software programs this field with the channel number corresponding to the analog input pin to be sampled and converted. See Channel assignment for details.								

Table 30-129. MESSAGE_TAG description

MESSAGE_TAG[0:3]	MESSAGE_TAG Meaning
0b0000	Result is sent to RFIFO 0
0b0001	Result is sent to RFIFO 1
0b0010	Result is sent to RFIFO 2
0b0011	Result is sent to RFIFO 3
0b0100	Result is sent to RFIFO 4

Table continues on the next page...

**Table 30-129. MESSAGE_TAG description
(continued)**

MESSAGE_TAG[0:3]	MESSAGE_TAG Meaning
0b0101	Result is sent to RFIFO 5
0b0110 - 0b0111	Reserved
0b1000	Null Message Received
0b1001	Reserved for customer use ¹
0b1010	Reserved for customer use ²
0b1011 - 0b1111	Reserved

1. These messages are treated as null messages. Therefore, they must obey the format for incoming null messages and return valid BUSY0/1 fields. "

30.6.2.2.1.2 Conversion command format for alternate configurations

Figure 30-118 describes the format for conversion commands when interfacing with the on-chip ADCs in one of the 14 alternate configurations. An alternate configuration is selected when the lowest byte (bits 24-31) of the conversion command is set to a value in the range 0x08-0x15. Each value in this range selects one of the 14 alternate configuration (0x08 selects Alternate Configuration 1, 0x15 selects Alternate Configuration 14). In the alternate configurations, the conversion result can be routed to one of the RFIFOs and/or to the parallel side interface to communicate with an on-chip companion module, or to one or two on-chip companion modules. Bit fields in the corresponding Alternate Configuration Control Register and Extended Alternate Configuration Control Register select the Internal RFIFO or Parallel Side Interface as the destination for the conversion result. A time stamp information can be optionally requested.

All fields, except FFMT and ALT_CONFIG_SEL, are identical to the ones in the standard configuration format. Only the fields which are different from the standard format will be described here.

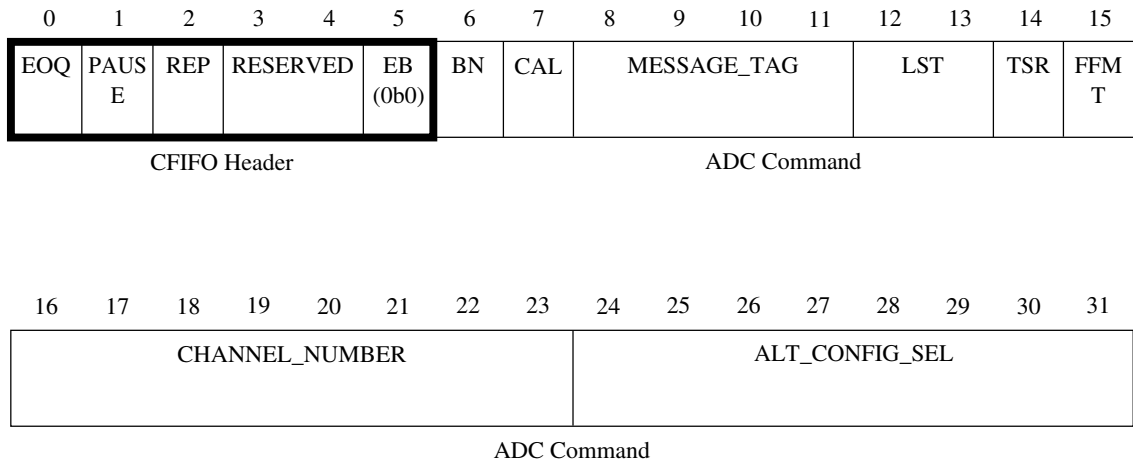


Figure 30-118. Conversion command format for alternate configurations

Table 30-130. Conversion command format for alternate configurations — field descriptions

Field	Description
EOQ	See Table 30-128
PAUSE	See Table 30-128
REP	See Table 30-128
EB	See Table 30-128
BN	See Table 30-128
CAL	See Table 30-128
MESSAGE_TAG	See Table 30-128
LST	See Table 30-128
TSR	See Table 30-128
FFMT	Flush or Format The function of this bit depends on the DEST field of the Alternate Configuration Control Register. If DEST is equal to 0b000, then FFMT defines the format in which the 12-bit conversion result are stored in the RFIFOs. If DEST is not equal to 0b000, then the FFMT bit is used to send a flush (soft-reset) signal through the parallel side interface to the companion module addressed by the DEST field. In case DEST is not equal to 0b000, the FMTA bit in the Alternate Configuration Control register is used to define the conversion result format. Note: The flush signal can be asserted along with a valid conversion result. In this case the companion module should execute the software-reset first and then consider the conversion result as a valid data for the filtering algorithm.
	0 Conversion Result Format set to right justified unsigned if DEST is equal to 0b000. No flush signal is sent through the side interface if DEST is not equal to 0b000.
	1 Conversion Result Format set to right justified signed if DEST is equal to 0b000. A flush signal is sent through the side interface if DEST is not equal to 0b000.
CHANNEL_NUMBER[0:7]	See Table 30-128

Table continues on the next page...

Table 30-130. Conversion command format for alternate configurations — field descriptions (continued)

Field	Description
ALT_CONFIG_SEL[]	Alternate Configuration Selection This field selects one of the alternate configurations.
	0x08 Alternate configuration 1
	0x09 Alternate configuration 2
	0x0A Alternate configuration 3
	0x0B Alternate configuration 4
	0x0C Alternate configuration 5
	0x0D Alternate configuration 6
	0x0E Alternate configuration 7
	0x0F Alternate configuration 8
	0x10 Alternate configuration 9
	0x11 Alternate configuration 10
	0x12 Alternate configuration 11
	0x13 Alternate configuration 12
	0x14 Alternate configuration 13
	0x15 Alternate configuration 14

30.6.2.2.1.3 Write configuration command format for on-chip ADC operation

Figure 30-119 describes the command message format for a write configuration command when interfacing with the on-chip ADCs. A write configuration command is used to set the control registers of the on-chip ADCs. No conversion data will be returned for a write configuration command. Write configuration commands are differentiated from read configuration commands by a negated R/W bit.

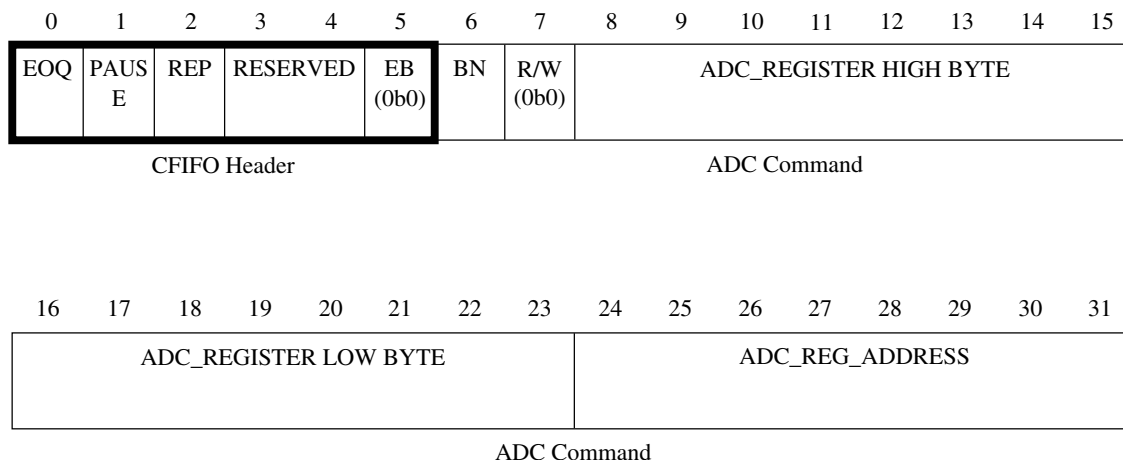
**Figure 30-119. Write configuration command format for on-chip ADC operation**

Table 30-131. Write configuration command format for on-chip ADC operation — field descriptions

Field	Description
EOQ	End Of Queue Bit
PAUSE	Pause Bit
REP	Repeat/loop Start Point Indication Bit
BN	Buffer Number Bit See Conversion command format for the standard configuration .
R/W	Read/Write bit A negated R/W indicates a write configuration command. 0 Write
ADC_REGISTER_HIGH_BYTE[0:7]	ADC Register High Byte Field REGISTER_HIGH_BYTE is the value to be written into the most significant 8 bits of control/configuration register when the R/W bit is negated.s
ADC_REGISTER_LOW_BYTE[0:7]	ADC Register Low Byte Field REGISTER_LOW_BYTE is the value to be written into the least significant 8 bits of a control/configuration register when the R/W bit is negated.
ADC_REG_ADDRESS[0:7]	ADC Register Address The ADC_REG_ADDRESS field selects a register on the ADC register set to be written or read. Only half-word addresses can be used.

30.6.2.2.1.4 Read configuration command format for on-chip ADC operation

Figure 30-120 describes the command message format for a read configuration command when interfacing with the on-chip ADCs. A read configuration command is used to read the contents of the on-chip ADC registers which are only accessible via command messages. Read configuration commands are differentiated from write configuration commands by an asserted R/W bit.

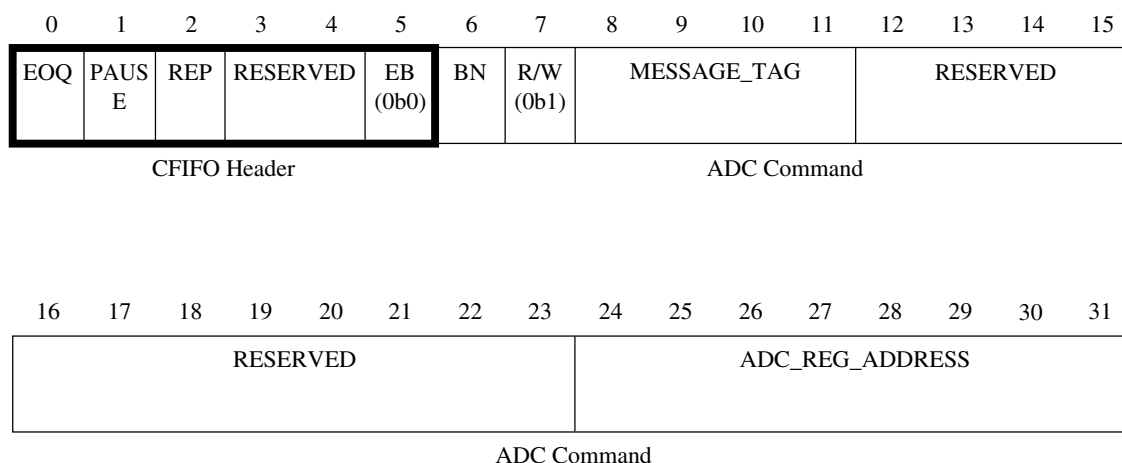


Figure 30-120. Read configuration command format for on-chip ADC operation

Table 30-132. Read configuration command format for on-chip ADC operation — field descriptions

Field	Description
EOQ	End Of Queue Bit
PAUSE	Pause Bit
REP	Repeat/loop Start Point Indication Bit
BN	Buffer Number Bit See Conversion command format for the standard configuration .
R/W	Read/Write bit An asserted R/W bit indicates a read configuration command. 1 Read
MESSAGE_TAG[0:3]	MESSAGE_TAG Field See Conversion command format for the standard configuration..
ADC_REG_ADDRESS[0:7]	ADC Register Address The ADC_REG_ADDRESS field selects a register on the ADC register set to be written or read. Only half-word addresses can be used.

30.6.2.2.1.5 ADC result format for on-chip ADC operation

When the *FIFO Control Unit* receives a return data message, it decodes the MESSAGE_TAG field and stores the 16-bit data into the appropriate RFIFO. This section describes the *ADC result* portion of the *result message* returned by the on-chip ADCs. The 16-bit data stored in the RFIFOs can be:

- Data read from an ADC register with a read configuration command. In this case, the stored 16-bit data corresponds to the contents of the ADC register that was read.
- A time stamp. In this case, the stored 16-bit data is the value of the time base counter latched when the EQADC detects the end of the analog input voltage sampling. For details see [Time stamp feature](#).
- A conversion result, coming directly from the ADCs. In this case, the stored 16-bit data contains a right justified 14-bit result data. The conversion result can be calibrated or not depending on the status of CAL bit in the command that requested the conversion⁴. When the CAL bit is negated, this 14-bit data is obtained by executing a 2-bit left-shift on the 12-bit data resultant from the resolution adjustment on the 8 or 10 or 12-bit data received from the ADC. The resolution adjustment consists of changing the conversion result input from 8, 10 or 12 bits right aligned to a 12-bit word left aligned - see [ADC resolution selection feature](#) for details. When the CAL bit is asserted, this 14-bit data is the result of the calculations performed in

4. In case the conversion result is routed through an on-chip DSP via side interface, the calibration is applied before the data is sent to the DSP.

the EQADC MAC unit using the 12-bit data result of the resolution adjustment and the calibration constants GCC and OCC, or ALTGCC and ALTOCC - see [ADC calibration feature](#) for details. Then, this 14-bit data is further formatted into a 16-bit format according to the status of the FMT bit in conversion command of the standard configuration or FFMT bit in the conversion command of the alternate configurations⁵. When FMT/FFMT is asserted, the 14-bit result data is reformatted to look as if it was measured against an imaginary ground at VREF/2 (the MSB bit of the 14-bit result is inverted), and is sign-extended to a 16-bit format as in [Figure 30-121](#). When FMT/FFMT is negated, the EQADC zero-extends the 14-bit result data to a 16-bit format as in [Figure 30-122](#). Correspondence between the analog voltage in a channel and the calculated digital values is shown in [Table 30-134](#).

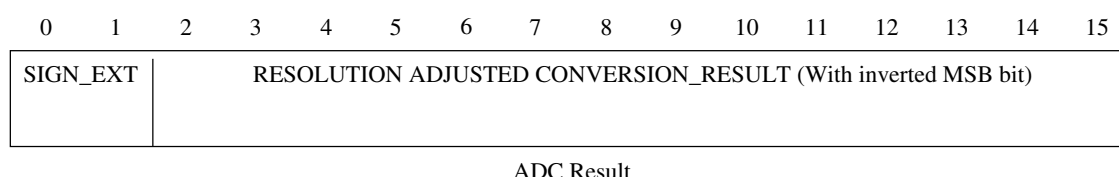


Figure 30-121. ADC Result Format when FMT=1 (Right Justified Signed)

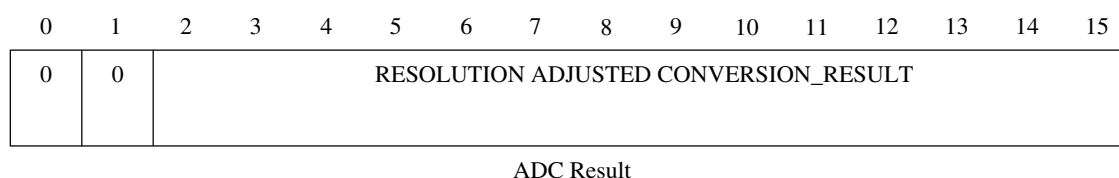


Figure 30-122. ADC result format when FMT=0 (right justified unsigned)

Table 30-133. ADC result formats — field descriptions

Field	Description
SIGN_EXT[0:1]	Sign Extension field SIGN_EXT only has meaning when FMT is asserted. SIGN_EXT is 0b00 when CONVERSION_RESULT is positive, and 0b11 when CONVERSION_RESULT is negative.
CONVERSION_RESULT[0:13]	Conversion Result field CONVERSION_RESULT is a digital value corresponding to the analog input voltage in a channel when the conversion command was initiated. The two's complement representation is used to express negative values.

In [Table 30-134](#) :

- VREF=VRH-VRL=5.12V. Resulting in one 12-bit count (LSB) =1.25mV.
- The two's complement representation is used to express negative values.

5. For simplicity, the following text will refer to FMT only, but when using alternate configurations, see [Conversion command format for alternate configurations](#).

Table 30-134. Correspondence between analog voltages and digital values

Conversions	Voltage Level on Channel (V)	Corresponding 8-bit Conversion Result Returned by the ADC	Corresponding 10-bit Conversion Result Returned by the ADC	Corresponding 12-bit Conversion Result Returned by the ADC	16-bit Result Sent to RFIFOs (FMT=0) ¹	16-bit Result Sent to RFIFOs (FMT=1) ¹
Single-Ended Conversions	5.12	-	-	0xFFF	0x3FFC	0x1FFC
		-	0x3FF	-	0x3FF0	0x1FF0
		0xFF	-	-	0x3FC0	0x1FC0
	5.12 - LSB	-	-	0xFFF	0x3FFC	0x1FFC
		-	0x3FF	-	0x3FF0	0x1FF0
		0xFF	-	-	0x3FC0	0x1FC0
	...	...	...	...	...	...
	2.56	-	-	0x800	0x2000	0x0000
		-	0x200	-	0x2000	0x0000
		0x80	-	-	0x2000	0x0000
	...	...	...	...	...	...
	1 LSB	-	-	0x001	0x0004	0xE004
		-	0x001	-	0x0010	0xE010
		0x01	-	-	0x0040	0xE040
	0	0x00	0x000	0x000	0x0000	0xE000
Differential Conversions	2.56	-	-	0xFFF	0x3FFC	0x1FFC
		-	0x3FF	-	0x3FF0	0x1FF0
		0xFF	-	-	0x3FC0	0x1FC0
	2.56 - LSB	-	-	0xFFF	0x3FFC	0x1FFC
		-	0x3FF	-	0x3FF0	0x1FF0
		0xFF	-	-	0x3FC0	0x1FC0
	...	...	...	...	...	...
	0	-	-	0x800	0x2000	0x0000
		-	0x200	-	0x2000	0x0000
		0x80	-	-	0x2000	0x0000
	...	...	...	...	...	...
	2.56 - LSB	-	-	0x001	0x0004	0xE004
		-	0x001	-	0x0010	0xE010
		0x01	-	-	0x0040	0xE040
	-2.56	0x00	0x000	0x000	0x0000	0xE000

1. Assuming uncalibrated conversion results.

30.6.3 Command/result queues

The Command and Result queues (CQueues and RQueues) are actually part of the EQADC system although they are not hardware implemented inside the EQADC. Each CQueue entry is a 32-bit Command Message. The last entry of a CQueue has the EOQ bit asserted to indicate that it is the last entry of the CQueue. RQueue entry is a 16-bit data.

See [Overview and basic terminology](#) for a description of the message formats and their flow in EQADC.

See [CQueue and RQueues usage](#) for examples of how CQueues and RQueues can be used.

30.6.4 EQADC command FIFOs

30.6.4.1 CFIFO basic functionality

There are six prioritized CFIFOs located in the EQADC. Each CFIFO is four entries deep, except CFIFO0 that can be configured to eight entries deep in extended mode, and each CFIFO entry is 32 bits long. A CFIFO serves as a temporary storage location for the command messages stored on the CQueues in the system memory. When a CFIFO is not full, the EQADC sets the corresponding CFFF bit in [EQADC FIFO and Interrupt Status Register \(EQADC_FISR_n\)](#). If CFFE is asserted in [EQADC Interrupt and DMA Control Register 0 \(EQADC_IDCR0\)](#), the EQADC generates requests for more commands from a CQueue. An interrupt request, served by the host CPU, is generated when CFFS is negated, and a DMA request, served by the DMAC, is generated when CFFS is asserted. The host CPU or the DMAC respond to these requests by writing to the [EQADC CFIFO Push Register \(EQADC_CFPR_n\)](#) to fill the CFIFO.

Note

The DMAC should be configured to write a single command (32-bit data) to the CFIFO push registers for every asserted DMA request it acknowledges. See [EQADC/DMAC interface](#) for DMAC configuration guidelines.

Note

CFIFO0 can be configured to work in an alternative way called Streaming Mode. This mode is very different from the mode described here because it maintains some stored commands to execute them several times in sequence and in loop.

Note

Only whole words must be written to EQADC_CFPR. Writing half-words or bytes to EQADC_CFPR will still push the whole 32-bit CF_PUSH field into the corresponding CFIFO, but undefined data will fill the areas of CF_PUSH that were not specifically designated as target locations for writing.

Figure 30-123 describes the important components in the CFIFO. Each CFIFO is implemented as a circular set of registers to avoid the need to move all entries at each push/pop operation. The Push Next Data Pointer points to the next available CFIFO location for storing data written into the EQADC Command FIFO Push Register. The Transfer Next Data Pointer points to the next entry to be removed from CFIFO_x when it completes a transfer. The *CFIFO Transfer Counter Control Logic* counts the number of entries in the CFIFO and generates DMA or interrupt requests to fill the CFIFO. TNXTPTR in [EQADC FIFO and Interrupt Status Register \(EQADC_FISR_n\)](#) indicates the index of the entry that is currently being addressed by the Transfer Next Data Pointer, and CFCTR, in the same register, provides the number of entries stored in the CFIFO. Using TNXTPTR and CFCTR, the absolute addresses for the entries indicated by the Transfer Next Data Pointer and by the Push Next Data Pointer can be calculated using the following formulas:

Transfer Next Data Pointer Address = CFIFO_x_BASE_ADDRESS + TNXTPTR_x*4

Push Next Data Pointer Address = CFIFO_x_BASE_ADDRESS + [(TNXTPTR_x + CFCTR_x) mod CFIFO_DEPTH] * 4

where

- $a \bmod b$ returns the remainder of the division of a by b .
- CFIFO_x_BASE_ADDRESS is the smallest memory mapped address allocated to a CFIFO_x entry.
- CFIFO_DEPTH is the number of entries contained in a CFIFO - four in this implementation.

When CFS_x in [EQADC CFIFO Status Register \(EQADC_CFSR\)](#) is TRIGGERED, the EQADC generates the proper control signals for the transfer of the entry pointed by Transfer Next Data Pointer. CFUF_x in [EQADC FIFO and Interrupt Status Register \(EQADC_FISR_n\)](#) is set when a CFIFO_x underflow event occurs. A CFIFO underflow occurs when the CFIFO is in TRIGGERED state and it becomes empty. No commands will be transferred from an underflowing CFIFO, nor will command transfers from lower priority CFIFOs be blocked. CFIFO_x is empty when the Transfer Next Data Pointer *x* equals the Push Next Data Pointer *x* and CFCTR_x is zero. CFIFO_x is full when the Transfer Next Data Pointer *x* equals the Push Next Data Pointer *x* and CFCTR_x is not zero.

When the EQADC completes the transfer of an entry from CFIFO_x: the transferred entry is popped from CFIFO_x, the CFIFO counter CFCTR in the [EQADC FIFO and Interrupt Status Register \(EQADC_FISR_n\)](#) is decremented by one, and Transfer Next Data Pointer *x* is incremented by one (or wrapped around) to point to the next entry in the CFIFO. The transfer of entries bound for the on-chip ADCs is considered completed when they are stored in the appropriate CBuffer.

When the EQADC_CFPR_x is written and CFIFO_x is not full, the CFIFO counter CFCTR_x is incremented by one, and the Push Next Data Pointer *x* then is incremented by one (or wrapped around) to point to the next entry in the CFIFO.

When the EQADC_CFPR_x is written but CFIFO_x is full, the EQADC will not increment the counter value and will not overwrite any entry in CFIFO_x.

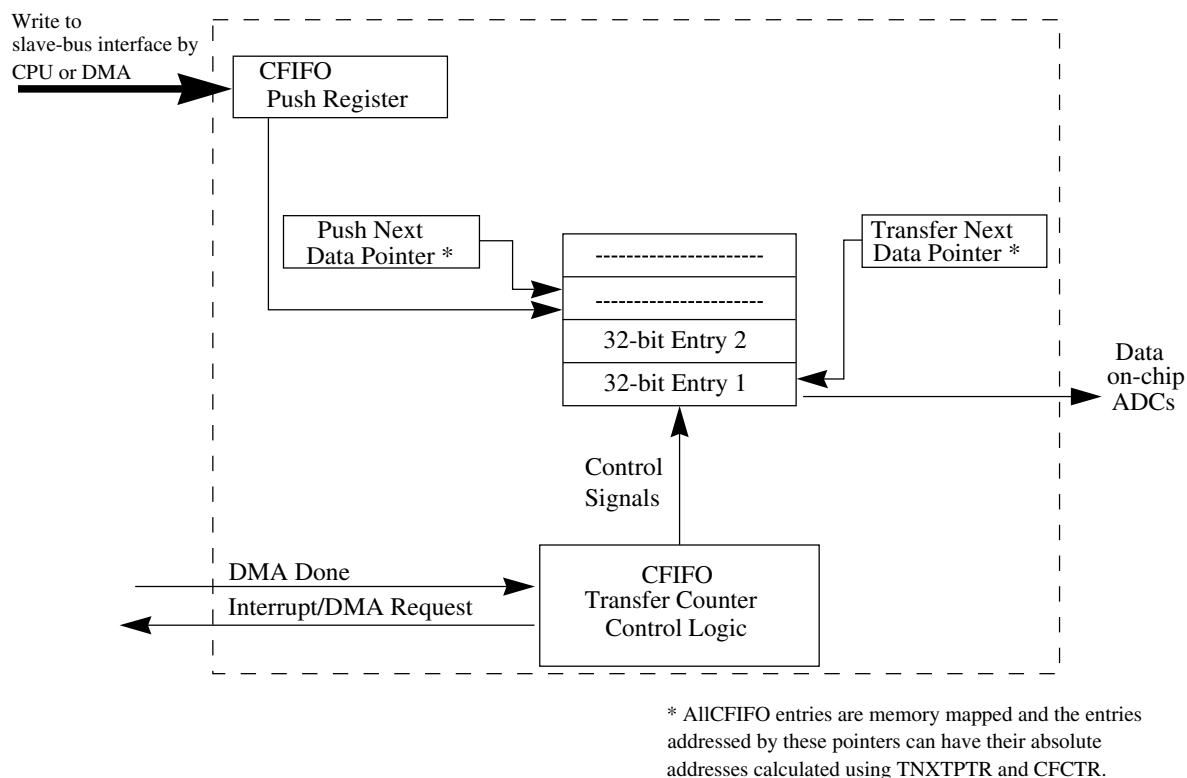


Figure 30-123. CFIFO Diagram

The detailed behavior of the Push Next Data Pointer and Transfer Next Data Pointer is described in the example shown in [Figure 30-124](#) where a CFIFO with 16 entries is shown for clarity of explanation, the actual hardware implementation has only four entries. In this example, CFIFOx with 16 entries is shown in sequence after pushing and transferring entries.

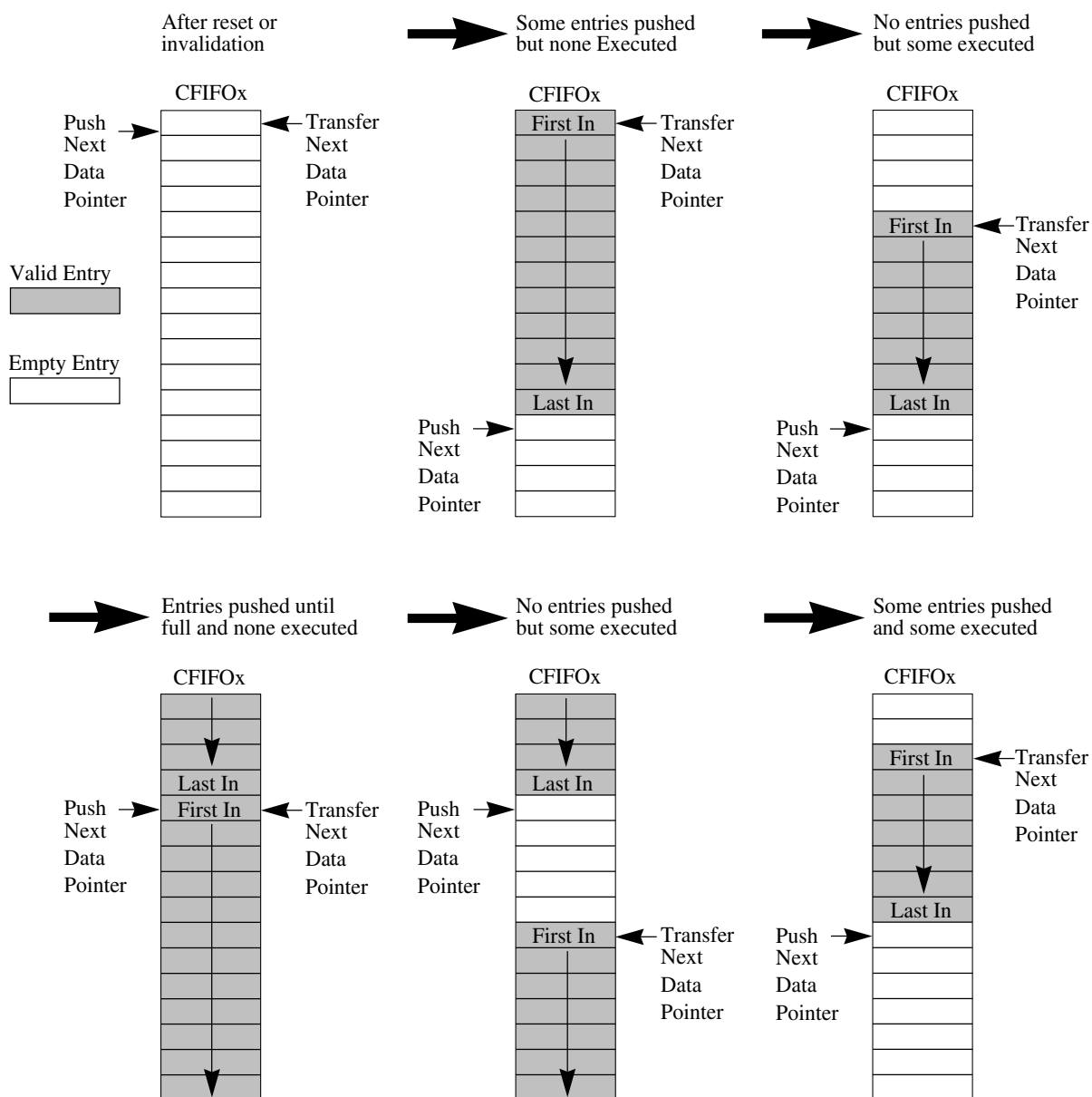


Figure 30-124. CFIFO Entry Pointer Example

30.6.4.2 CFIFO0 streaming mode description

CFIFO0 can be configured to operate in streaming mode to allow repetition of a group of commands several times without the need of refilling the registers as in the normal mode of operation of CFIFOs. This mode makes use of the additional bit in the Conversion Command Word (CCW) called 'Repeat' (REP bit). The purpose of this bit is to mark in the command queue, where to start a repeating sequence. This location is stored in an additional pointer 'Repeat Pointer'.

Streaming mode requires 2 trigger inputs. The standard queue 0 trigger, in this mode referred to as Repeat Trigger and a new internal trigger input to the eQADC called Advance Trigger (no filter available).

CFIFO0 is configured to operate in streaming mode by setting the bit STRME0 as described in [EQADC CFIFO Control Register 0 \(EQADC_CFCR0\)](#). CFIFO0 is eight entries deep in extended mode by setting the bit CFEEE0 in the same EQADC_CFCR register, and each entry is 32 bits long. This CFIFO0 serves as a local storage of a few commands that need to be executed sequentially as in a FIFO but can contain sub-queues that need to be executed several times. The CFFF0 bit in [EQADC FIFO and Interrupt Status Register \(EQADC_FISR_n\)](#) is used to assure the CFIFO0 is not full and command messages are stored from address 0x0 to 0x7.

30.6.4.2.1 CFIFO0 operation in streaming mode

In Streaming mode, the CFIFO0 is filled with CCWs using the DMA exactly the same as existing modes. The CFIFO executes commands as per the existing modes until it executes a Conversion Command Word with the Repeat bit set. When this CCW is executed, the Repeat Pointer is set to point to this FIFO location and from this CCW onwards, CFIFO0 entries is not invalidated, that is, the Repeat Pointer prevents this and subsequent entries from being overwritten.

The queue continues to execute until a CCW with an asserted Pause bit is completed; then the queue stops and enters the Pause state, waiting for a trigger. This is the same as normal behavior.

The Pause state is exited in one of two ways: Repeat Trigger or Repeat Trigger with Advance Trigger. The Repeat trigger with no Advance trigger causes the Transfer Next Data Pointer to be loaded with the Repeat Pointer location and CCWs are then executed from the Repeat Pointer back to the Pause bit. This means that a section of the CFIFO0 is repeatedly executed every time a Repeat Trigger occurs.

The Repeat trigger with the Advance trigger pending causes all CCWs from the Repeat pointer to the Pause bit to be invalidated and the CCW after the pause bit to be executed. This is achieved by invalidating the Repeat Pointer. The effect is that the queue advances beyond the repeating section of the CFIFO0 to execute new CCWs.

Note that the Advance trigger can occur at any time between Repeat triggers, but is only granted when the next Repeat trigger occurs. Prior to that it is pending.

In a typical application, the queue is made of some configuration commands to the ADC (to flush the decimator or turn on pad pull-up/down) followed by a repeating section of ADC conversions on one or more ADC channels from one or more sensors; followed by a few more configuration commands; then more repeating ADC conversions, until the

entire engine cycle is complete; when the queue is restarted. The mechanism described permits any number of repeating sub-queues to be loaded and executed, interspersed by configuration commands.

30.6.4.2.2 Triggering description in streaming mode

The additional trigger signal ATRIG0 is detected by a separate circuit that is configured by the bit field AMODE0 as described in [EQADC CFIFO Control Register 0 \(EQADC_CFCR0\)](#). This trigger signal is used as an advance control of pop pointer of CFIFO0. In addition, it is used as the enable trigger for the Repeat trigger. This means it is necessary to have an Advance trigger first to enable the detection of the Repeat trigger. When the Repeat trigger is enabled, the Advance trigger is used to advance the pop pointer beyond some loop sub-queue. And it is used to disable the Repeat trigger by executing a Pause without a previous REP bit. [Figure 30-125](#) and [Figure 30-126](#) illustrate these two triggers dependency.

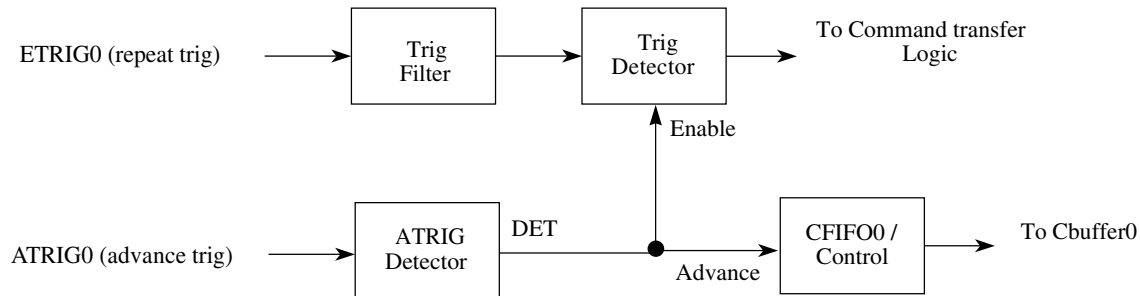


Figure 30-125. Advance trigger and repeat trigger relationship Block Diagram

In Streaming mode, the CFIFO0 is filled with CCWs using the DMA as usual. The two triggers are configured to positive edge and single scan mode.

The SSS bit is asserted and the trigger detector of the Repeat trigger is disabled in the start of the queue. It is necessary to receive the first Advance trigger to enable the detector of the other trigger. This enable is useful when the Repeat trigger is received all the time and the trigger signal can be disabled when it is not desired.

The Advance trigger is received and detected and the Repeat trigger detector is enabled. No commands are executed until now.

The Repeat trigger is detected and the commands start to be executed in sequence. If a REP bit is decoded with the PAUSE bit, the loop is configured and the CFIFO0 commands stop to be executed. The next Repeat trigger is waited to start the execution of the loop again, or the Advance trigger can be detected to break the loop and advance the queue in CFIFO0. The Repeat trigger detector remains enabled.

If the Advance trigger is received and the next command in the CFIFO0 does not present the REP bit set, this means the CFIFO0 is not starting a new loop. In this case (outside a loop) if a PAUSE bit is decoded, this means to disable the Repeat trigger detector. This can be useful if the Repeat trigger is not required for some interval of time. The Repeat trigger detector is enabled again when the next Advance trigger event is detected.

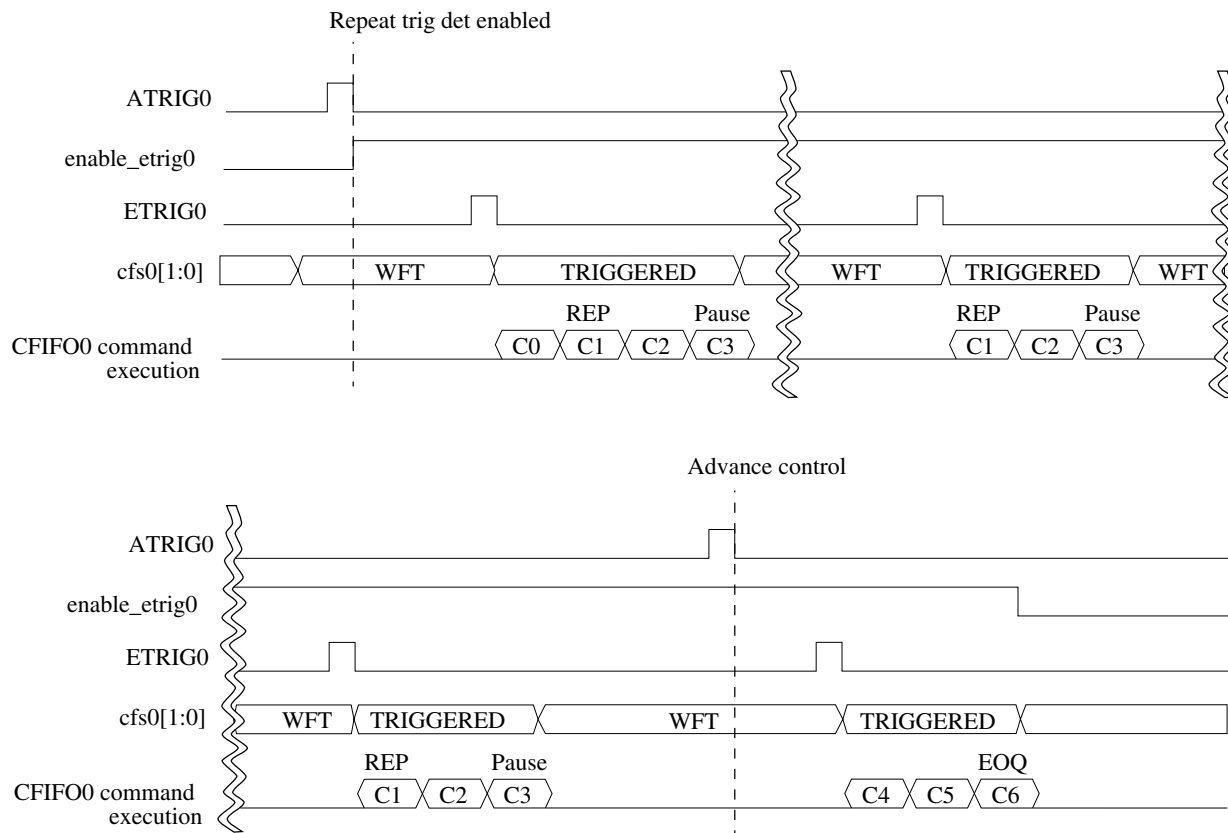


Figure 30-126. Timing diagram of typical use of Advance trigger and Repeat trigger

30.6.4.2.3 CFIFO0 diagram description in streaming mode

Figure 30-127 represents the main components of CFIFO0 in streaming mode. However, some signals behave in a different way from the common operation. The Push Next Data Pointer points to the next available CFIFO0 location for storing data written into the EQADC Command FIFO Push Register. The Transfer Next Data Pointer points to the next entry to be transferred to Cbuffer. The Repeat Pointer points to the first entry of the repeating sub-queue. TNXTPTR in [EQADC FIFO and Interrupt Status Register \(EQADC_FISR_n\)](#) indicates the index of the entry that is currently being addressed by the Transfer Next Data Pointer, and CFCTR, in the same register, provides the number of entries stored in the CFIFO.

When CFS0 in [EQADC CFIFO Status Register \(EQADC_CFSR\)](#) is TRIGGERED, the EQADC generates the proper control signals for the transfer of the entry pointed by Transfer Next Data Pointer. CFUF0 in [EQADC FIFO and Interrupt Status Register \(EQADC_FISR_n\)](#) is set when CFIFO0 underflow event occurs. A CFIFO underflow occurs when the CFIFO is in TRIGGERED state and it is empty. No commands will be transferred from an underflowing CFIFO, nor will command transfers from lower priority CFIFOs be blocked. CFIFO0 is empty when CFCTR0 is zero. CFIFO0 is full when (CFCTR0 mod CFIFO_DEPTH) is zero but CFCTR0 is not zero.

When the EQADC completes the transfer of an entry from CFIFO0 in loop condition: the transferred entry is not popped from CFIFO0, the CFIFO counter CFCTR in the [EQADC FIFO and Interrupt Status Register \(EQADC_FISR_n\)](#) is not decremented by one, and Transfer Next Data Pointer 0 is incremented by one (or wrapped around) to point to the next entry in the CFIFO0.

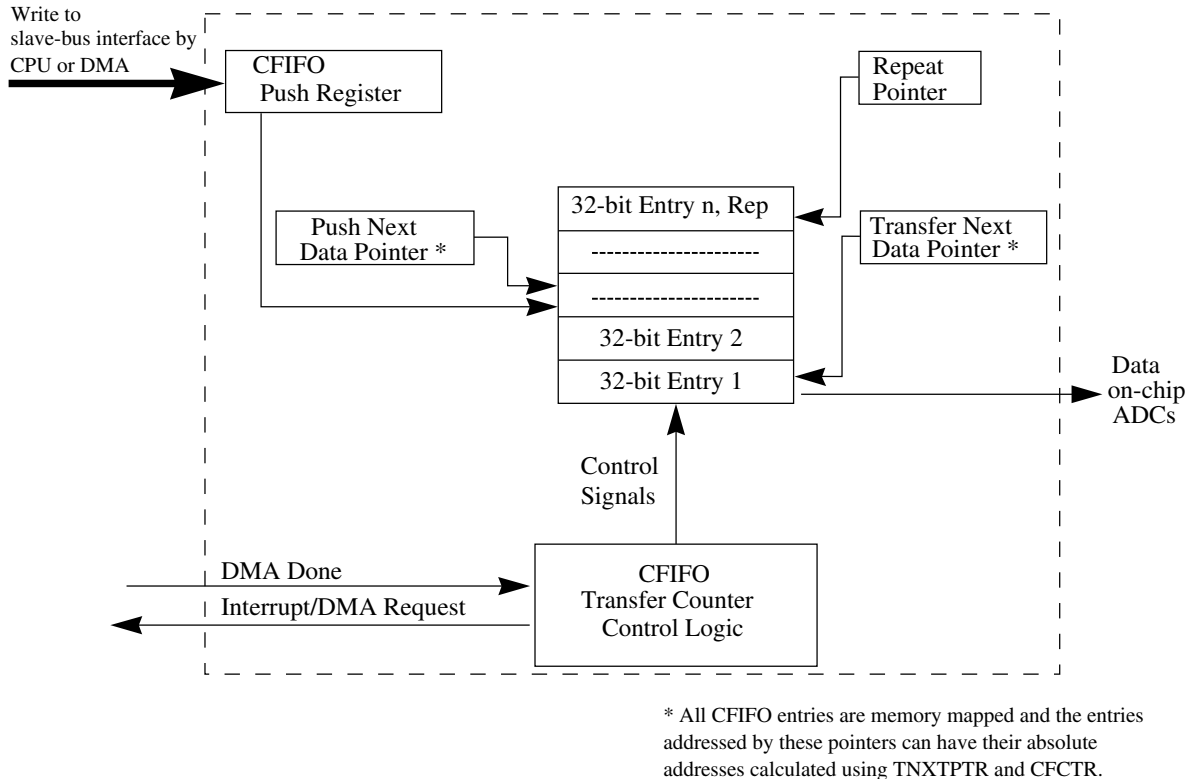
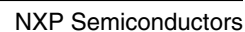


Figure 30-127. CFIFO0 in streaming mode diagram

The detailed behavior of the Push Next Data Pointer and Transfer Next Data Pointer is described in the example shown in [Figure 30-128](#) where a CFIFO with 16 entries is shown for clarity of explanation, the actual hardware implementation has only four/eight entries. In this example, CFIFO0 with 16 entries is shown in sequence after pushing and transferring entries.



MPC5777C Reference Manual, Rev. 8, 11/2016

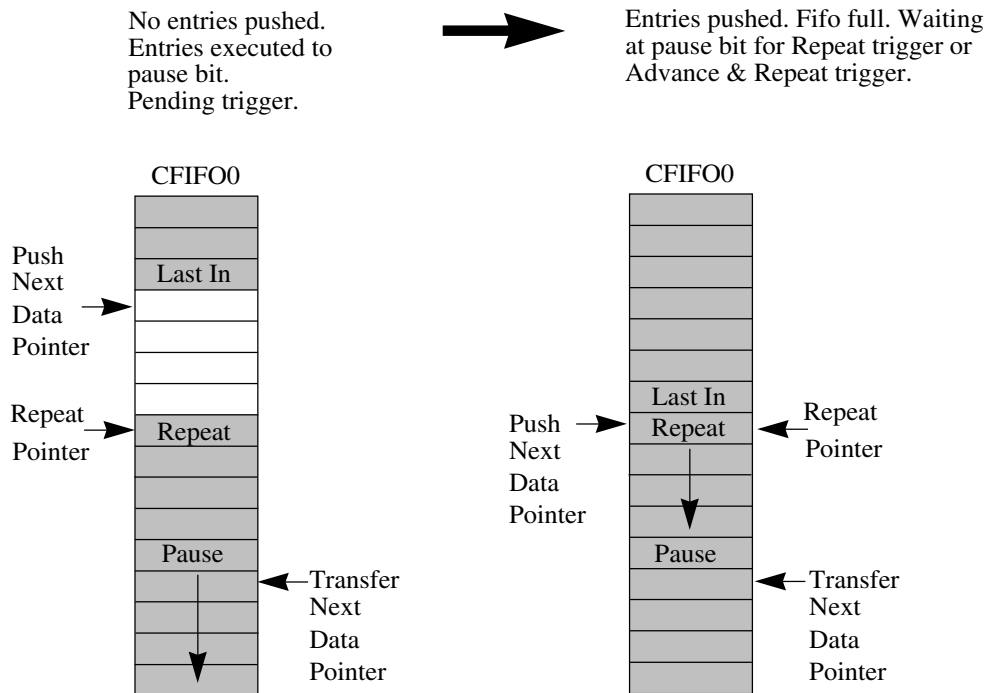


Figure 30-129. CFIFO0 in streaming mode entry pointer example (continued)

30.6.4.2.4 Streaming mode error conditions

In the repeat state, the existing error conditions still apply, but now there are new ways to trigger them. Now, the CCWs are not being invalidated so the DMA is not able to load more CCWs into those locations. So a queue overflow becomes more likely, and occurs if the repeat loop is longer than 8 entries. If all CCWs in the CFIFO0 are executed and no Pause bit or EOQ bit is detected, the eQADC will signal an underflow error. In practice this may limit a repeating queue to 7 entries since otherwise an underflow will occur at the point a Repeat with Advance trigger occurs, and there is no command in the CFIFO0 to execute. The exception is a final command with both a Pause and an EOQ bit set. The End of Queue bit EOQ continues to operate as in normal mode, unless the Repeat mode is enabled. In this case the Pause bit takes precedence and a Repeat trigger causes the jump back described. A Repeat trigger with Advance trigger causes the queue to end.

Another error condition occur when the repeat trigger is in the TRIGGERED state and a new repeat trigger is received. In this case, a trigger overflow occurs but the CFIFO0 is defined to not restart the loop. The trigger in this case is not used in the CFIFO0, but the overflow is indicated.

30.6.4.3 CFIFO common prioritization and command transfer

The CFIFO priority is fixed according to the CFIFO number. A CFIFO with a smaller number has a higher priority. When commands of distinct CFIFOs are bound for the same destination (CBuffer), the higher priority CFIFO is always served first. A TRIGGERED, not-underflowing CFIFO will start the transfer of its commands when:

- its commands are bound for an internal CBuffer that is not full, and it is the highest priority triggered CFIFO sending commands to that CBuffer.

A triggered CFIFO with commands bound for a certain CBuffer consecutively transfers its commands to it until:

- an asserted End Of Queue bit is reached, or;
- an asserted Pause bit is encountered and the CFIFO is configured for edge trigger mode, or;
- CFIFO is configured for level trigger mode and a closed gate is detected, or;
- in case its commands are bound for an internal CBuffer, a higher priority CFIFO that uses the same internal CBuffer is triggered

The prioritization logic of the EQADC, depicted in [Figure 30-130](#), is composed of three independent sub-blocks: one prioritizing CFIFOs with commands bound for CBuffer0, another prioritizing CFIFOs with commands for CBuffer1. As these two sub-blocks are independent, simultaneous writes to CBuffer0, to CBuffer1 are allowed. The hardware identifies the destination of a command by decoding the BN bit in the command message - see [Message format in EQADC](#) for details.

Note

Triggered but empty CFIFOs, underflowing CFIFOs, are not considered for prioritization. No data from these CFIFOs will be sent to the CBuffers and nor will they stop lower priority CFIFOs from transferring commands.

Whenever CBuffer0 is able to receive new entries, the prioritization sub-block selects the highest-priority triggered CFIFO with a command bound for CBuffer0, and writes its command into the buffer. In case CBuffer0 is able to receive new entries but there are no triggered CFIFOs with commands bound for it, nothing is written to the buffer. The sub-block prioritizing CBuffer1 usage behaves in the same way.

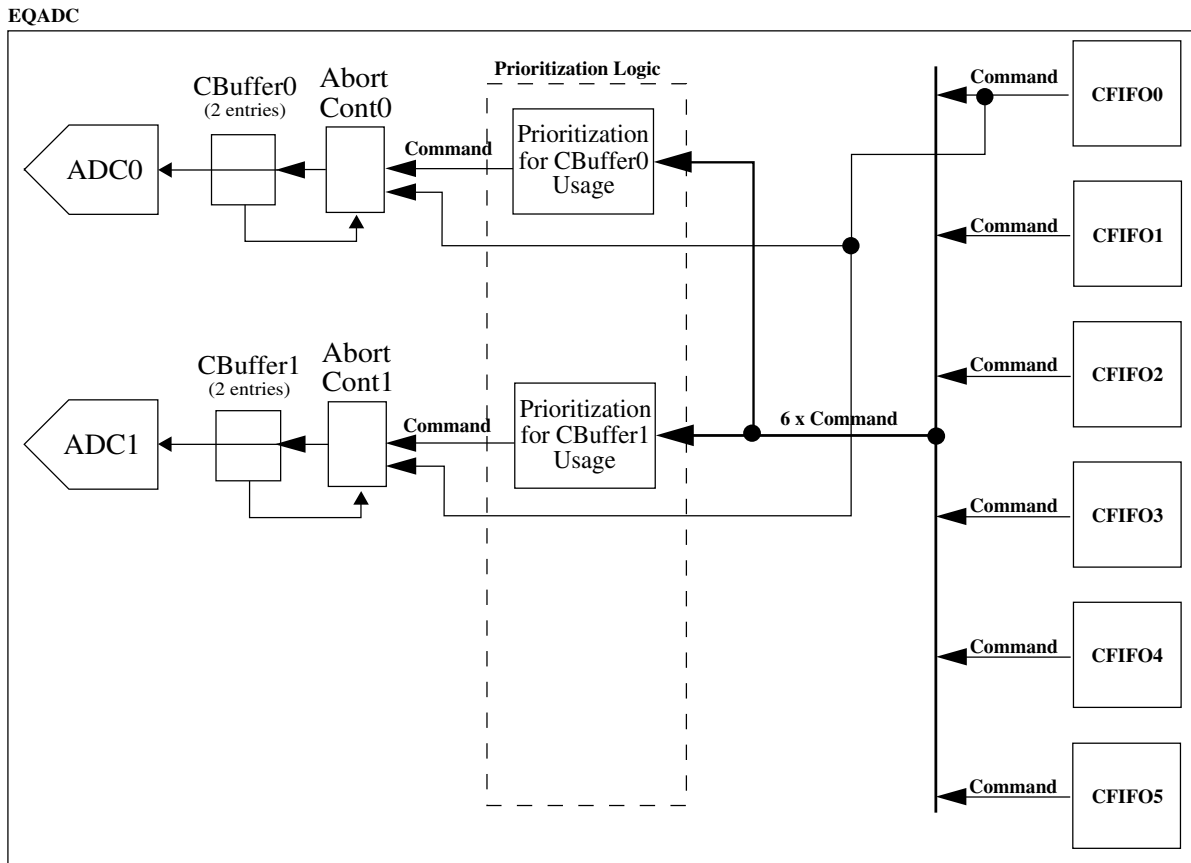


Figure 30-130. CFIFO prioritization logic

30.6.4.4 CFIFO prioritization in abort mode

The CFIFO priority does not change when the EQADC is configured to allow abortion of conversion execution in on-chip ADC analog blocks. However, CFIFO0 is the only one that can be enabled to abort conversions.

This feature is necessary when the timing of some conversion is very important. In normal priority scheme, when CFIFO0 is triggered, its conversion command can be put behind 2 pending conversion commands in the Cbuffer due to the queue structure. Considering that these 2 pending commands are from lower priority CFIFOs and that the delay between the trigger and the sampling of the command from CFIFO0 can be unacceptable, EQADC can be configured to permit immediate conversion commands from CFIFO0 with abort function.

When CFIFO0 is triggered and abort is enabled, up to 2 commands in Cbuffer0 or Cbuffer1 are stored in a side register. The abort request signal is generated to ADC0 or ADC1 and the confirmation of ADC reset/ready is waited to send the command from CFIFO0 to the decoded Cbuffer.

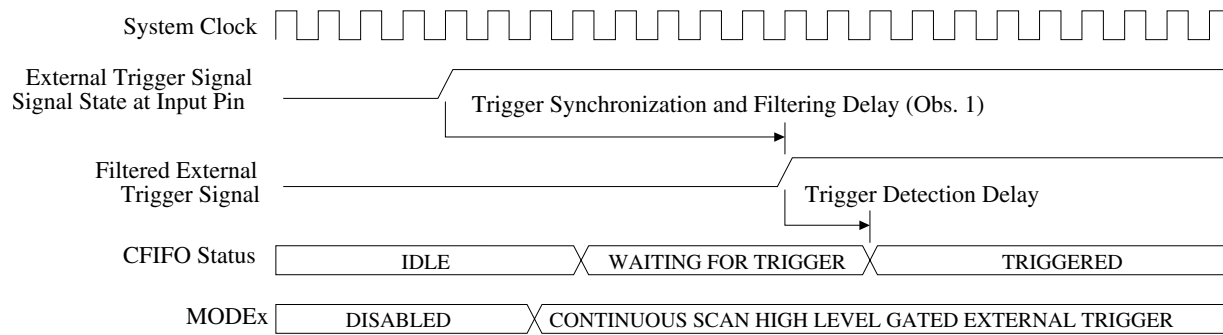
After the transfer of all commands from CFIFO0, the recovery phase restores the up to 2 commands that were in Cbuffer when the abort occurred. After this recovery phase, it is established the normal process of prioritization of commands from CFIFOs.

30.6.4.5 External trigger event detection

The digital filters for trigger signals can be individually bypassed by asserting the input control signals `eqadc_intern_trig_sel5-0`. When the filter is bypassed, the ETRIG input signal is not filtered and the logic after the filter receives a copy of this input trigger signal.

The Digital Filter Length field in [EQADC External Trigger Digital Filter Register \(EQADC_ETDFR\)](#) specifies the minimum number of system clocks that the ETRIG0-5 signals must be held at a logic level to be recognized as valid. All ETRIG signals are filtered. A counter for each queue trigger is implemented to detect a transition between logic levels. The counter counts at the system clock rate. The corresponding counter is cleared and restarted each time the signal transitions between logic levels. When the corresponding counter matches the value specified by the Digital Filter Length field in [EQADC External Trigger Digital Filter Register \(EQADC_ETDFR\)](#), the EQADC considers the ETRIG logic level to be valid and passes that new logic level to the rest of the EQADC.

The filter is only for filtering the ETRIG signal. Logic after the filter checks for transitions between filtered values, such as for detecting the transition from a filtered logic level zero to a filter logic level one in rising edge external trigger mode. The EQADC can detect rising edge, falling edge, or level gated external triggers. The digital filter will always be active independently of the status of the MODEx field in [EQADC CFIFO Control Register 0 \(EQADC_CFCR0\)](#), but the edge, level detection logic is only active when MODEx is set to a value different from disabled, and in case MODEx is set to single scan mode, when the SSS bit is asserted. Note that the time necessary for an external trigger event to result into a CFIFO status change is not solely determined by the DFL field in the [EQADC External Trigger Digital Filter Register \(EQADC_ETDFR\)](#). After being synchronized to the system clock and filtered, a trigger event is checked against the CFIFO trigger mode. Only then, after a valid trigger event is detected, the EQADC accordingly changes the CFIFO status. See [Figure 30-131](#) for an example.



Obs.

- 1: This delay is about 2 clocks when the filter bypass control is asserted.

Figure 30-131. ETRIG event propagation example

30.6.4.6 CFIFO scan trigger modes

The EQADC supports two different scan modes, single-scan and continuous-scan. Refer to [Table 30-135](#) for a summary of these two scan modes. When a CFIFO is triggered, the EQADC scan mode determines whether the EQADC will stop command transfers from a CFIFO, and wait for software intervention to rearm the CFIFO to detect new trigger events, upon detection of an asserted EOQ bit in the last transfer. See [Message format in EQADC](#) for details about command formats.

CFIFOs can be configured in single-scan or continuous-scan mode. When a CFIFO is configured in single-scan mode, the EQADC scans the CQueue one time. The EQADC stops future command transfers from the triggered CFIFO after detecting the EOQ bit set in the last transfer. After a EOQ bit is detected, software involvement is required to rearm the CFIFO so that it can detect new trigger events.

When a CFIFO is configured for continuous-scan mode, no software involvement is necessary to rearm the CFIFO to detect new trigger events after an asserted EOQ is detected. In continuous-scan mode the whole CQueue is scanned multiple times.

The EQADC also supports different triggering mechanisms for each scan mode. The EQADC will not transfer commands from a CFIFO until the CFIFO is triggered. The combination of scan modes and triggering mechanisms allows the support of different requirements for scanning input channels. The scan mode and trigger mechanism are configured by programming the MODEx field in [EQADC CFIFO Control Register 0 \(EQADC_CFCR0\)](#).

Enabled CFIFOs can be triggered by software or external trigger events. The elapsed time from detecting a trigger to transferring a command is a function of clock frequency, trigger synchronization, trigger filtering or not, programmable trigger events, command

transfer, CFIFO prioritization, CBuffer availability, etc. Fast and predictable transfers can be achieved by ensuring that the CFIFO is not underflowing and that the target CBuffer is not full when the CFIFO is triggered.

30.6.4.6.1 Disabled Mode

The MODEx field in [EQADC CFIFO Control Register 0 \(EQADC_CFCR0\)](#) for all of the CFIFOs can be changed from any other mode to disabled at any time. No trigger event can initiate command transfers from an CFIFO which has its MODE field programmed to disabled.

Note

If MODEx is not disabled, it must not be changed to any other mode besides disabled. If MODEx is disabled and the CFIFO status is IDLE, MODEx can be changed to any other mode.

If MODEx is changed to disabled:

- The CFIFO execution status will change to IDLE. The timing of this change depends on whether a command is being transferred or not:
 - When no command transfer is in progress, the EQADC switches the CFIFO to IDLE status immediately.
 - When a command transfer to an on-chip CBuffer is in progress, the EQADC will complete the transfer, update TC_CF, and switch CFIFO status to IDLE. Command transfers to the internal CBuffers are considered completed when a command is written to the buffers.
- The CFIFOs are not invalidated automatically. The CFIFO still can be invalidated by writing a "1" to the CFINVx bit in [EQADC CFIFO Control Register 0 \(EQADC_CFCR0\)](#). Certify that CFS has changed to IDLE before setting CFINVx.
- The TC_CFx value also is not reset automatically, but it can be reset by writing "0" to it.
- The SSS bit in [EQADC FIFO and Interrupt Status Register \(EQADC_FISRn\)](#) is negated. The SSS bit can be set even if a "1" is written to the SSE bit in [EQADC CFIFO Control Register 0 \(EQADC_CFCR0\)](#) in the same write that the MODEx field is changed to a value other than disabled.
- The trigger detection hardware is reset. If MODEx is changed from disabled to an edge trigger mode, a new edge, matching that edge trigger mode, is needed to trigger the command transfers from the CFIFO.

Note

CFIFO fill requests, which generated when CFFF is asserted, are not automatically halted when MODEx is changed to disabled. CFIFO fill requests will still be generated until CFFE is cleared in [EQADC Interrupt and DMA Control Register 0 \(EQADC_IDCR0\)](#).

30.6.4.6.2 Single-scan mode

In single-scan mode, a single pass through a sequence of command messages in a CQueue is performed.

In single-scan software trigger mode, the CFIFO is triggered by an asserted Single-Scan Status bit (SSS) in [EQADC FIFO and Interrupt Status Register \(EQADC_FISR_n\)](#). The SSS bit is set by writing "1" to the Single-Scan Enable bit (SSE) in [EQADC CFIFO Control Register 0 \(EQADC_CFCR0\)](#).

In single-scan edge- or level-trigger mode, the respective triggers are only detected when the SSS bit is asserted. When the SSS bit is negated, all trigger events for that CFIFO are ignored. Writing a "1" to the SSE bit can be done during the same write cycle that the CFIFO operation mode is configured.

Only the EQADC can clear the SSS bit. Once SSS is asserted, it remains asserted until the EQADC completes the CQueue scan, or the CFIFO operation mode (MODEx) in [EQADC CFIFO Control Register 0 \(EQADC_CFCR0\)](#) is changed to disabled. The SSSx bit will be negated while MODEx is disabled.

30.6.4.6.2.1 Single-Scan Software Trigger

When single-scan software trigger mode is selected, the CFIFO is triggered by an asserted SSS bit. The SSS bit is asserted by writing "1" to the SSE bit. Writing to SSE while SSS is already asserted will not have any effect on the state of the SSS bit, nor will it cause a trigger overrun event.

The CFIFO commands start to be transferred when the CFIFO becomes the highest priority CFIFO using a not-full on-chip CBuffer. When an asserted EOQ bit is encountered, the EQADC will clear the SSS bit. Setting the SSS bit is required for the EQADC to start the next scan of the queue.

The Pause bit has no effect in single-scan software trigger mode.

30.6.4.6.2.2 Single-scan edge trigger

When SSS is asserted and an edge triggered mode is selected for a CFIFO, an appropriate edge on the associated trigger signal causes the CFIFO to become TRIGGERED. For example, if rising-edge trigger mode is selected, the CFIFO becomes TRIGGERED when a rising edge is sensed on the trigger signal. The CFIFO commands start to be transferred when the CFIFO becomes the highest priority CFIFO using a not-full on-chip CBuffer.

When an asserted EOQ bit is encountered, the EQADC clears SSS and stops command transfers from the CFIFO. An asserted SSS bit and a subsequent edge trigger event are required to start the next scan for the CFIFO. When an asserted Pause bit is encountered, the EQADC stops command transfers from the CFIFO, but SSS remains set. Another edge trigger event is required for command transfers to continue. A trigger overrun happens when the CFIFO is in TRIGGERED state and an edge trigger event is detected.

30.6.4.6.2.3 Single-scan level trigger

When SSS is asserted and a level gated trigger mode is selected, the input level on the associated trigger signal puts the CFIFO in TRIGGERED state. When the CFIFO is asserted to high-level gated trigger, a high level signal opens the gate, and a low level closes the gate. When the CFIFO is set to low-level gated trigger mode, a low level signal opens the gate, and a high level closes the gate. If the corresponding level is already present, setting the SSS bit triggers the CFIFO. The CFIFO commands start to be transferred when the CFIFO becomes the highest priority CFIFO using a not-full on-chip CBuffer.

The EQADC clears the SSS bit and stops transferring commands from a TRIGGERED CFIFO when an asserted EOQ bit is encountered or when CFIFO status changes from TRIGGERED due to the detection of a closed gate. If a closed gate is detected while no command transfers are taking place and the CFIFO status is TRIGGERED, the CFIFO status is immediately changed to IDLE, the SSS bit is negated, and the PF flag is asserted. Once the transmission is completed, the TC_CF counter is updated, the SSS bit is negated, the PF flag is asserted, and the CFIFO status is changed to IDLE. An asserted SSS bit and a level trigger are required to restart the CFIFO. Command transfers will restart from the point they have stopped.

The Pause bit has no effect in single-scan level-trigger mode.

30.6.4.6.3 Continuous-scan mode

In continuous-scan mode, multiple passes looping through a sequence of command messages in a CQueue are executed. When a CFIFO is programmed for a continuous-scan mode, the SSE bit in the [EQADC CFIFO Control Register 0 \(EQADC_CFCR0\)](#) does not have any effect.

30.6.4.6.3.1 Continuous-scan software trigger

When a CFIFO is programmed to continuous-scan software trigger mode, the CFIFO is triggered immediately. The CFIFO commands start to be transferred when the CFIFO becomes the highest priority CFIFO using a not-full on-chip CBuffer. When a CFIFO is programmed to run in continuous-scan software trigger mode, the EQADC will not halt transfers from the CFIFO until the CFIFO operation mode is modified to disabled or a higher priority CFIFO preempts it. Although command transfers will not stop upon detection of an asserted EOQ bit, the EOQF is set and, if enabled, an EOQ interrupt request is generated.

The Pause bit has no effect in continuous-scan software trigger mode.

30.6.4.6.3.2 Continuous-scan edge trigger

When rising, falling, or either edge trigger mode is selected for a CFIFO, a corresponding edge on the associated ETRIG signal places the CFIFO in TRIGGERED state. The CFIFO commands start to be transferred when the CFIFO becomes the highest priority CFIFO using a not-full on-chip CBuffer.

When an EOQ or a Pause is encountered, the EQADC halts command transfers from the CFIFO and, if enabled, the appropriate interrupt requests are generated. Another edge trigger event is required to resume command transfers but no software involvement is required to rearm the CFIFO in order to detect such event.

A trigger overrun happens when the CFIFO is already in TRIGGERED state and a new edge trigger event is detected.

30.6.4.6.3.3 Continuous-scan level trigger

When high or low level gated trigger mode is selected, the input level on the associated trigger signal places the CFIFO in TRIGGERED state. When high-level gated trigger is selected, a high-level signal opens the gate, and a low level closes the gate. The CFIFO commands start to be transferred when the CFIFO becomes the highest priority CFIFO.

using a not-full on-chip CBuffer. Although command transfers will not stop upon detection of an asserted EOQ bit at the end of a command transfer, the EOQF is asserted and, if enabled, an EOQ interrupt request is generated.

The EQADC stops transferring commands from a TRIGGERED CFIFO when CFIFO status changes from TRIGGERED due to the detection of a closed gate. If a closed gate is detected while no command transfers are taking place and the CFIFO status is TRIGGERED, the CFIFO status is immediately changed to WAITING FOR TRIGGER and the PF flag is asserted. Once the transmission is completed, the TC_CF counter is updated, the PF flag is asserted, and the CFIFO status is changed to WAITING FOR TRIGGER. Command transfers will restart as the gate opens.

The Pause bit has no effect in continuous-scan level-trigger mode.

30.6.4.6.4 CFIFO scan trigger mode start/stop summary

Table 30-135 summarizes the start and stop conditions of command transfers from CFIFOs for all of the single-scan and continuous-scan trigger modes.

Table 30-135. CFIFO scan trigger mode - command transfer start/stop summary

Trigger mode	Requires asserted SSS to recognize trigger events?	Command transfer start/restart condition	Stop on asserted EOQ bit ¹ ?	Stop on asserted Pause bit ² ?	Other command transfer stop condition ^{3,4}
Single Scan Software	Don't Care	Asserted SSS bit.	Yes	No	None.
Single Scan Edge	Yes	A corresponding edge occurs when the SSS bit is asserted.	Yes	Yes	None.
Single Scan Level	Yes	Gate is opened when the SSS bit is asserted.	Yes	No	EQADC also stops transfers from the CFIFO when CFIFO status changes from TRIGGERED due to the detection of a closed gate. ⁵
Continuous Scan Software	No	CFIFO starts automatically after being configured into this mode.	No	No	None.
Continuous Scan Edge	No	A corresponding edge occurs.	Yes	Yes	None.
Continuous Scan Level	No	Gate is opened.	No	No	EQADC also stops transfers from the CFIFO when CFIFO status changes from TRIGGERED due to the detection of a closed gate. ⁵

1. See [CQueue completion status](#) for more information on EOQ.

2. See [Pause status](#) for more information on Pause.
3. EQADC always stops command transfers from a CFIFO when the CFIFO operation mode is disabled.
4. EQADC always stops command transfers from a CFIFO when a higher priority CFIFO is triggered. See [CFIFO common prioritization and command transfer](#) for information on CFIFO priority.
5. If a closed gate is detected while no command transfers are taking place, it will have immediate effect on the CFIFO status.

30.6.4.7 CFIFO and trigger status

30.6.4.7.1 CFIFO operation status

Each CFIFOs has its own CFIFO status field. CFIFO status (CFS) can be read from [EQADC CFIFO Status Register \(EQADC_CFSR\)](#). [Figure 30-132](#) and [Table 30-136](#) indicate the CFIFO status switching condition. See [Table 30-43](#) for the meaning of each CFIFO operation status. The last CFIFO to transfer a command to an on-chip CBuffer can be read from the LCFTCB n ($n=0,1$) fields in [EQADC CFIFO Status Snapshot Register 0 \(EQADC_CFSSR0\)](#).

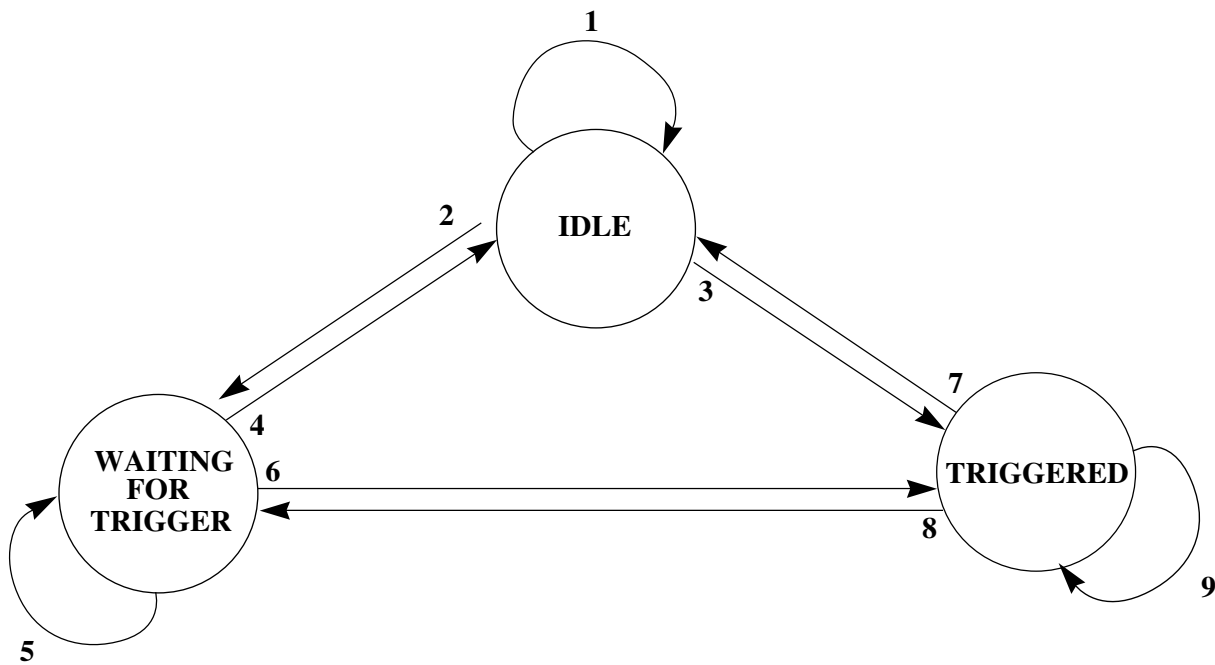


Figure 30-132. State machine of CFIFO status

Table 30-136. Command FIFO status switching condition

No.	From current CFIFO status (CFS)	To new CFIFO status (CFS)	Status switching condition
1	IDLE (00)	IDLE (0b00)	- CFIFO Mode is programmed to disabled, OR - CFIFO Mode is programmed to single-scan edge or level trigger mode and SSS is negated.
2		WAITING FOR TRIGGER (0b10)	- CFIFO Mode is programmed to continuous-scan edge or level trigger mode, OR - CFIFO Mode is programmed to single-scan edge or level trigger mode and SSS is asserted, OR - CFIFO Mode is programmed to single-scan software trigger mode.
3		TRIGGERED (0b11)	CFIFO Mode is programmed to continuous-scan software trigger mode
4	WAITING FOR TRIGGER (10)	IDLE (0b00)	CFIFO Mode is modified to disabled mode.
5		WAITING FOR TRIGGER (0b10)	No trigger occurred.
6		TRIGGERED (0b11)	- Appropriate edge or level trigger occurred, OR - CFIFO Mode is programmed to single-scan software trigger mode and SSS bit is asserted.

Table continues on the next page...

**Table 30-136. Command FIFO status switching condition
(continued)**

No.	From current CFIFO status (CFS)	To new CFIFO status (CFS)	Status switching condition
7	TRIGGERED (11)	IDLE (0b00)	- CFIFO in single-scan mode, EQADC detects the EOQ bit asserted at end of command transfer, and CFIFO Mode is not modified to disabled. OR - CFIFO, in single-scan level trigger mode, and the gate closes while no commands are being transferred from the CFIFO, and CFIFO Mode is not modified to disabled. OR - CFIFO, in single-scan level trigger mode, and EQADC detects a closed gated at end of command transfer, and CFIFO Mode is not modified to disabled. OR - CFIFO Mode is modified to disabled mode and CFIFO was not transferring commands. - CFIFO Mode is modified to disabled mode while CFIFO was transferring commands, and CFIFO completes or aborts the transfer.
8		WAITING FOR TRIGGER (0b10)	- CFIFO in single or continuous-scan edge trigger mode, EQADC detects the Pause bit asserted at the end of command transfer, the EOQ bit in the same command is negated, and CFIFO Mode is not modified to disabled, OR - CFIFO in continuous-scan edge trigger mode, EQADC detects the EOQ bit asserted at the end of command transfer, and CFIFO Mode is not modified to disabled, OR - CFIFO, in continuous-scan level trigger mode, and the gate closes while no commands are being transferred from the CFIFO, and CFIFO Mode is not modified to disabled, OR - CFIFO, in continuous-scan level trigger mode, and EQADC detects a closed gated at end of command transfer, and CFIFO Mode is not modified to disabled.
9		TRIGGERED (0b11)	No event to switch to IDLE or WAITING FOR TRIGGER status has happened.

30.6.4.7.2 CQueue completion status

The End of Queue Flag (EOQF) in [EQADC FIFO and Interrupt Status Register \(EQADC_FISR_n\)](#) is asserted when the EQADC completes the transfer of a CFIFO entry with an asserted EOQ bit. Software sets the EOQ bit in the last Command Message of a CQueue to indicate that this entry is the end of the CQueue - see [Message format in EQADC](#) for information on command message formats. The transfer of entries bound for the on-chip ADCs is considered completed when they are stored in the appropriate CBuffer.

The command with a EOQ bit asserted is valid and will be transferred. When EOQIE in [EQADC CFIFO Control Register 0 \(EQADC_CFCR0\)](#) and EOQF are asserted, the EQADC will generate an End of Queue interrupt request.

In single-scan modes, command transfers from the corresponding CFIFO will cease when EQADC completes the transfer of a entry with an asserted EOQ. Software involvement is required to rearm the CFIFO so that it can detect new trigger events.

Note

An asserted EOQFx only implies that EQADC has finished transferring a command with an asserted EOQ bit from CFIFOx. It does not imply that result data for the current command and for all previously transferred commands has been returned to the appropriate RFIFO.

30.6.4.7.3 Pause status

In edge trigger mode, when the EQADC completes the transfer of a CFIFO entry with an asserted Pause bit, the EQADC will stop future command transfers from the CFIFO and set the corresponding Pause Flag (PF) in [EQADC FIFO and Interrupt Status Register \(EQADC_FISR_n\)](#). See [Message format in EQADC](#) for information on command message formats. The EQADC ignores the Pause bit in command messages in any software and external level trigger mode. The EQADC sets the PF flag upon detection of an asserted Pause bit only in single or continuous-scan edge trigger mode. When the PF flag is set for a CFIFO in single-scan edge trigger mode, the SSS bit will not be cleared in [EQADC FIFO and Interrupt Status Register \(EQADC_FISR_n\)](#).

In level trigger mode, the definition of the PF flag has been redefined. In level trigger mode, when CFIFOx is in TRIGGERED status, PFx is set when CFIFO status changes from TRIGGERED due to detection of a closed gate. The pause flag interrupt routine can be used to verify if the a complete scan of the CQueue was performed. If a closed gate is detected while no command transfers are taking place, it will have immediate effect on the CFIFO status.

When PIE in [EQADC CFIFO Control Register 0 \(EQADC_CFCR0\)](#) and PF are asserted, the EQADC will generate a Pause interrupt request.

Note

In edge trigger mode, an asserted PFx only implies that the EQADC finished transferring a command with an asserted PAUSE bit from CFIFOx. It does not imply that result data for the current command and for all previously transferred commands has been returned to the appropriate RFIFO.

Note

In software or level trigger mode, when the EQADC completes the transfer of an entry from CFIFOx with an asserted Pause bit, PFX will not be set and command transfers will continue without pausing.

30.6.4.7.4 Trigger overrun status

When a CFIFO is configured for edge- or level-trigger mode and is in TRIGGERED state, an additional trigger occurring for the same CFIFO results in a trigger overrun. The trigger overrun bit for the corresponding CFIFO will be set ($TORFx = 1$) in [EQADC FIFO and Interrupt Status Register \(EQADC_FISR_n\)](#). When TORIE in [EQADC CFIFO Control Register 0 \(EQADC_CFCR0\)](#) and TORF are asserted, the EQADC generates a trigger overrun interrupt request.

Note

The trigger overrun flag will not set for CFIFOs configured for software trigger mode.

30.6.4.7.5 Command sequence non-coherency detection

The EQADC provides a mechanism to indicate if a command sequence has been completely executed without interruptions. A command sequence is defined as a group of consecutive commands bound for the same CBuffer and it is expected to be executed without interruptions. A command sequence is coherent if its commands are executed in order without interruptions. Since commands are stored in the CBuffers before being executed in the EQADC, a command sequence is coherent if, while it is transferring commands to a CBuffer, the buffer is only fed with commands from that sequence without ever becoming empty.

A command sequence starts when:

- a CFIFO in TRIGGERED state transfers its first command to CBuffer.
- the CFIFO is constantly transferring commands and the previous command sequence ended.
- the CFIFO resumes command transfers after being interrupted.

And a command sequence ends when:

- an asserted EOQ bit is detected on the last transferred command.

- CFIFO is in edge-trigger mode and asserted PAUSE bit is detected on the last transferred command.
- the CBuffer to which the next command is bound is different from the one to which the last command was transferred.

Figure 30-133 shows examples of how the EQADC would detect command sequences when transferring commands from a CFIFO to a CBuffer. The smallest possible command sequence can have a single command as shown in example 3 of Figure 30-133.

CQueue with a two command sequences

CF5_CB1_CM0
CF5_CB1_CM1
CF5_CB1_CM2
CF5_CB1_CM3 (Pause = 1)
CF5_CB1_CM4
CF5_CB1_CM5
CF5_CB1_CM6 (EOQ = 1)

Example 1

Assuming that these commands are transferred by a CFIFO configured for edge trigger mode and the command transfers are never interrupted, the EQADC would check for non-coherency of two command sequences: one formed by commands 0, 1, 2, 3, and the other by commands 4, 5, 6.

CQueue with a three command sequences

CF5_CB1_CM0
CF5_CB1_CM1
CF5_CB1_CM2
CF5_CP5_CB0_CM3
CF5_CB0_CM4
CF5_CB1_CM5
CF5_CB1_CM6 (EOQ = 1)

Example 2

Assuming that command transfers from the CFIFO are never interrupted, the EQADC would check for non-coherency of three command sequences. The first being formed by commands 0, 1, 2, the second by commands 3, 4 and the third by commands 5, 6. Note that even when the commands of this CQueue are transferred through a CFIFO in continuous-scan mode, the first three commands and the last two commands of this CQueue would still constitute two distinct command sequences, although they are all bound for the same CBuffer, since an asserted EOQ ends a command sequence.

CQueue with a seven command sequences

CF5_CB1_CM0
CF5_CB2_CM1
CF5_CB3_CM2
CF5_CF5_CB1_CM3
CF5_CB0_CM4
CF5_CB2_CM5
CF5_CB1_CM6 (EOQ = 1)

Example 3

The EQADC would check for non-coherency of seven command sequences, all containing a single command, but NCF would never get set.

CF_xCB_aCM_n - Command *n* in CFIFO_x bound for CBuffer_a

Figure 30-133. Command sequence examples

The NCF flag is used to indicate command sequence non-coherency. When the NCFx flag is asserted, it indicates that the command sequence being transferred through CFIFOx became non-coherent. The NCF flag only becomes asserted for CFIFOs in TRIGGERED state.

A command sequence is non-coherent when, after transferring the first command of a sequence from a CFIFO to a CBuffer, it cannot successively send all the other commands of the sequence before any of the following conditions are true:

- The CFIFO through which commands are being transferred is preempted by a higher priority CFIFO which sends commands to the same CBuffer. The NCF flag becomes asserted immediately after the first command transfer from the preempting CFIFO, that is the higher priority CFIFO, to the CBuffer in use is completed. See [Figure 30-134](#).

-

Once a command sequence starts to be transferred, the EQADC will check for the command sequence coherency until the command sequence ends or until one of the conditions below becomes true.

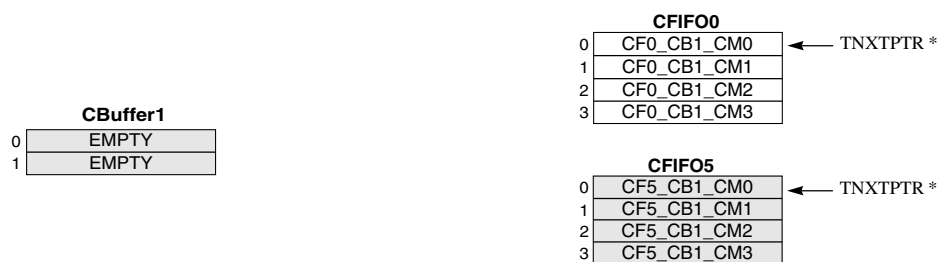
- The command sequence became non-coherent.
- The CFIFO status changed from TRIGGERED.
- The CFIFO underflew.

Once command transfers restart/continue, the non-coherency hardware will behave as if the command sequence started from that point. [Figure 30-135](#) depicts how the non-coherency hardware will behave when a non-coherency event is detected.

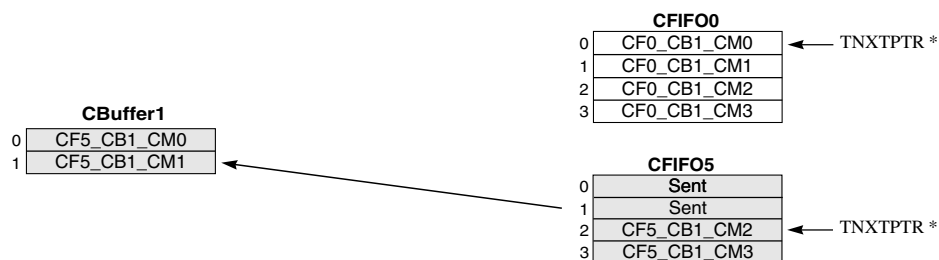
Note

If MODEx is changed to disabled while a CFIFO is transferring commands, the NCF flag for that CFIFO will not become asserted.

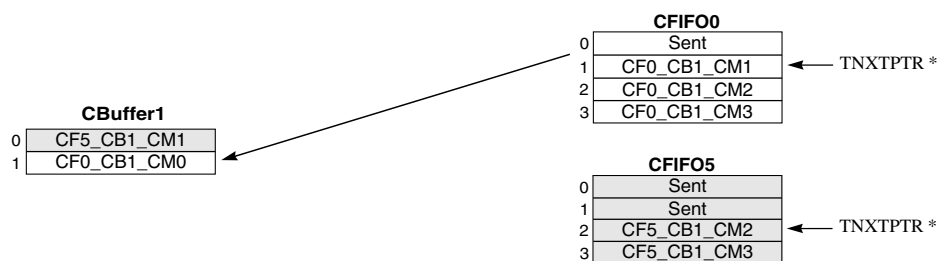
CFIFO and trigger status



(a) CFIFO0 and CFIFO5 both have commands to be sent to CBuffer1, and both are not triggered



(b) CFIFO5 becomes triggered and transfers two commands to CBuffer1



(c) CFIFO0 becomes triggered and transfers a command to CBuffer1. The sequence sent through CFIFO5 becomes non-coherent.

* TNXTPTR - Transfer Next Data Pointer

CF_x_CB_a_CM_n - Command *n* in CFIFO_x bound for CBuffer_a

Figure 30-134. Non-coherency event when different CFIFOs use the same CBuffer

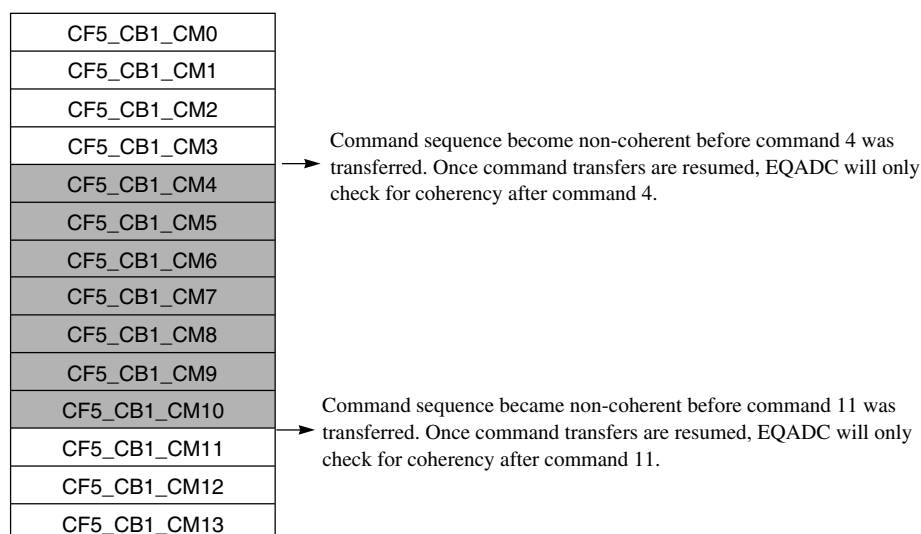


Figure 30-135. Non-coherency detection when transfers from a command sequence are interrupted

30.6.5 EQADC result FIFOs

30.6.5.1 RFIFO basic functionality

There are six RFIFOs located in the EQADC. Each RFIFO is four entries deep, and each RFIFO entry is 16 bits long. Each RFIFO serves as a temporary storage location for the one of the RQueues allocated in system memory. Result data is saved in the RFIFOs before being moved into the system RQueues. When an RFIFO is not empty, the EQADC sets the corresponding RFDF bit in [EQADC FIFO and Interrupt Status Register \(EQADC_FISR_n\)](#). If RFDE is asserted in [EQADC Interrupt and DMA Control Register 0 \(EQADC_IDCR0\)](#), the EQADC generates a request so that an RFIFO entry is moved to the RQueue. An interrupt request, served by the host CPU, is generated when RFDS is negated, and a DMA request, served by the DMAC, is generated when RFDS is asserted. The host CPU or the DMAC responds to these requests by reading [EQADC Result FIFO Pop Register \(EQADC_RFPR_n\)](#) to retrieve data from the RFIFO.

Note

The DMAC should be configured to read a single result (16-bit data) from the RFIFO pop registers for every asserted DMA request it acknowledges. See [EQADC/DMAC interface](#) for DMAC configuration guidelines.

Note

Reading a word, a half-word, or any bytes from EQADC_RFPRx will pop an entry from RFIFOx, and the RFCTRx field will be decremented by one.

Note

When the DMA is selected ((RFDSx=1), readings should be done only controlled by the DMA protocol. Other ways to flush the RFIFOx to empty the memory can cause a locked DMA request (permanent ON).

Figure 30-136 describes the important components in the RFIFO. Each RFIFO is implemented as a circular set of registers to avoid the need to move all entries at each push/pop operation. The Pop Next Data Pointer always points to the next RFIFO message to be retrieved from the RFIFO when reading EQADC_RFPR. The Receive Next Data Pointer points to the next available RFIFO location for storing the next incoming message from the on-chip ADCs. The *RFIFO Counter Logic* counts the number of entries in RFIFO and generates interrupt or DMA requests to drain the RFIFO.

POPNEXTPTR in [EQADC FIFO and Interrupt Status Register \(EQADC_FISR_n\)](#) indicates which entry is currently being addressed by the Pop Next Data Pointer, and RFCTR, in the same register, provides the number of entries stored in the RFIFO. Using POPNEXTPTR and RFCTR, the absolute addresses for Pop Next Data Pointer and Receive Next Data Pointer can be calculated using the following formulas:

Pop Next Data Pointer Address= RFIFOx_BASE_ADDRESS + POPNEXTPTRx*4

Receive Next Data Pointer Address = RFIFOx_BASE_ADDRESS + [(POPNEXTPTRx + RFCTRx) mod RFIFO_DEPTH] * 4

where

- $a \bmod b$ returns the remainder of the division of a by b .
- RFIFOx_BASE_ADDRESS is the smallest memory mapped address allocated to an RFIFOx entry.
- RFIFO_DEPTH is the number of entries contained in a RFIFO - four in this implementation.

When a new message arrives and RFIFOx is not full, the EQADC copies its contents into the entry pointed by the Receive Next Data Pointer. The RFIFO counter RFCTRx in [EQADC FIFO and Interrupt Status Register \(EQADC_FISR_n\)](#) is incremented by one, and the Receive Next Data Pointer x is also incremented by one (or wrapped around) to

point to the next empty entry in RFIFOx. However, if the RFIFOx is full, the EQADC sets the RFOF in [EQADC FIFO and Interrupt Status Register \(EQADC_FISR_n\)](#). The RFIFOx will not overwrite the older data in the RFIFO, the new data will be ignored, and the Receive Next Data Pointer x is not incremented or wrapped around. RFIFOx is full when the Receive Next Data Pointer x equals the Pop Next Data Pointer x and RFCTR_x is not zero. RFIFOx is empty when the Receive Next Data Pointer x equals the Pop Next Data Pointer x and RFCTR_x is zero.

When the EQADC RFIFO Pop Register x is read and the RFIFOx is not empty, the RFIFO counter RFCTR_x is decremented by one, and the POP Next Data Pointer is incremented by one (or wrapped around) to point to the next RFIFO entry.

When the EQADC RFIFO Pop Register x is read and RFIFOx is empty, EQADC will not decrement the counter value and the POP Next Data Pointer x will not be updated. The read value will be undefined.

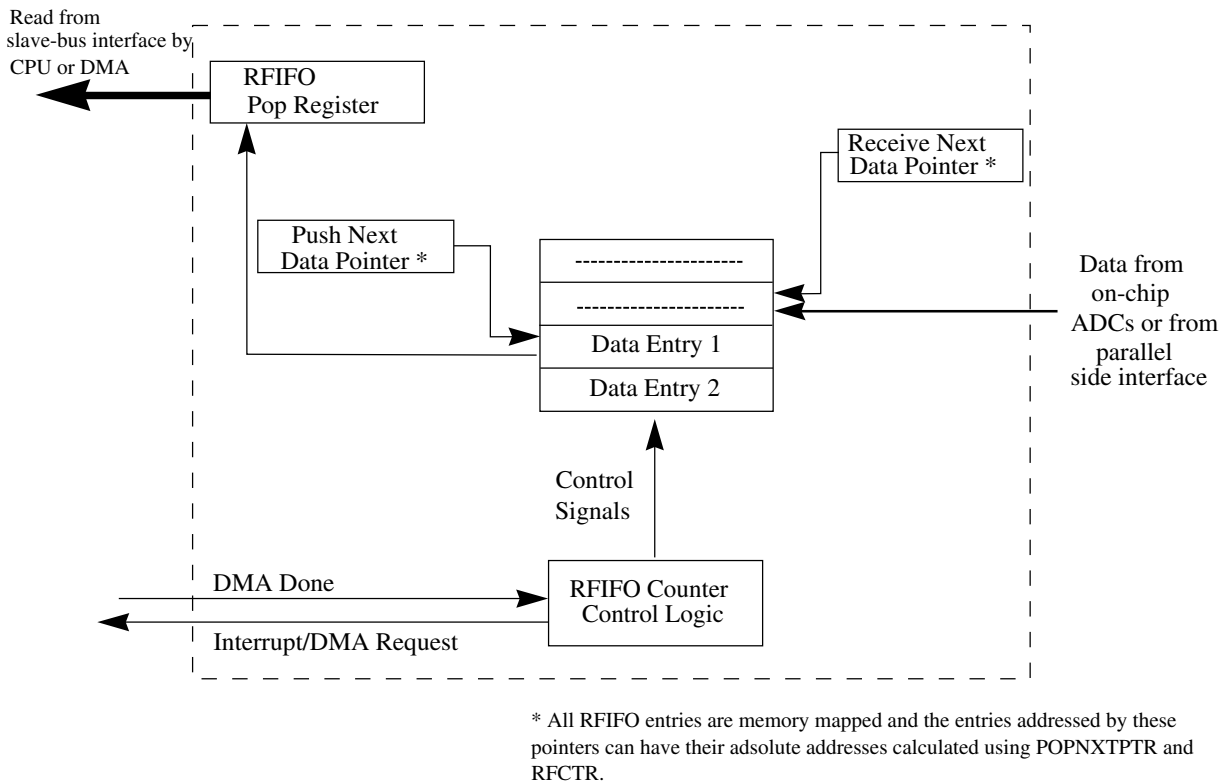


Figure 30-136. RFIFO diagram

The detailed behavior of the Pop Next Data Pointer and Receive Next Data Pointer is described in the example shown in [Figure 30-137](#) where an RFIFO with 16 entries is shown for clarity of explanation, the actual hardware implementation has only four entries. In this example, RFIFOx with 16 entries is shown in sequence after popping or receiving entries.

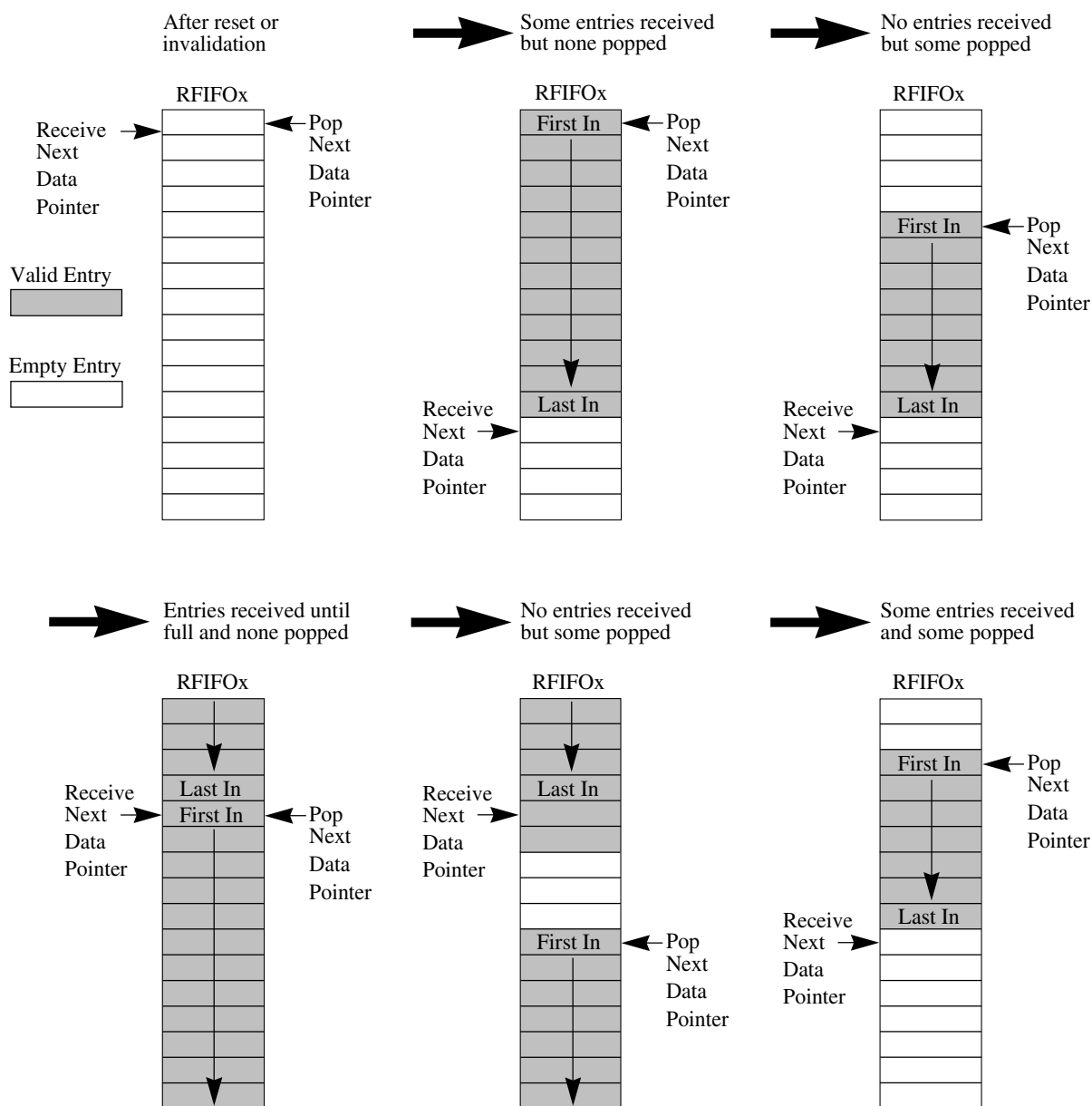


Figure 30-137. RFIFO entry pointer example

30.6.5.2 Distributing result data into RFIFOs

Data to be moved into the RFIFOs can come from different sources: from ADC0, from ADC1, or from the on-chip companion module through the PSI. All result data comes with the MESSAGE_TAG / MESSAGE_TAG2 field and the DEST / DEST2 field defining what should be done with the received data. The EQADC hardware decodes the MESSAGE_TAG / MESSAGE_TAG2 and DEST/DEST2 fields and:

- stores the 16-bit data into the appropriate RFIFO if the MESSAGE_TAG / MESSAGE_TAG2 indicates a valid RFIFO number, or;
- sends the 16-bit data, the MESSAGE_TAG / MESSAGE_TAG2 and the DEST / DEST2 data through the PSI to an on-chip companion module (as a decimation filter), or;
- ignores the data in case of a null or "reserved for customer use" MESSAGE_TAG / MESSAGE_TAG2.

In general received data is moved into RFIFOs as they become available, while an exception happens when multiple results from different sources become available at the same time. In that case, result data from ADC0 is processed first, result data from ADC1 is only processed after all ADC0 data is processed, and finally returned data from companion module is only processed after all data from ADC0/1 is processed.

When time-stamped results return from the on-chip ADCs, the conversion result and the time stamp are always moved to the RFIFOs in consecutive clock cycles in order to guarantee they are always stored in consecutive RFIFO entries.

30.6.6 On-chip ADC configuration and control

30.6.6.1 Enabling and disabling the on-chip ADCs

The on-chip ADCs have an enable bit (ADC0/1_EN) in [ADC0/1 Control Registers \(ADC0_CR and ADC1_CR\)](#) that allows the enabling of the ADCs only when necessary. When the enable bit for an ADC is negated, the clock input to that ADC is stopped. The ADCs are disabled out of reset - ADC0/1_EN bits are negated - to allow for their safe configuration. The ADC must only be configured when its enable bit is negated. Once the enable bit of an ADC is asserted, clock input to is started.

Note

Conversion commands sent to the CBuffer of a disabled ADC are ignored by the ADC control hardware.

Note

Due to legacy reasons, the EQADC will always wait 120 ADC clocks before issuing the first conversion command following the enabling of one of on-chip ADCs, or the exiting of stop mode. There are two independent counters checking for this delay: one clocked by ADC0_CLK and another by

ADC1_CLK. Conversion commands can start to be executed whenever one of these counters completes counting 120 ADC clocks.

30.6.6.2 ADC clock and conversion speed

The clock input to the ADCs is defined by setting the ADC0/1_ODD_PS, the ADC0/1_CLK_SEL and the ADC0/1_CLK_PS fields in [ADC0/1 Control Registers \(ADC0_CR and ADC1_CR\)](#). When the ADC0/1_CLK_SEL is set, the ADC clock frequency is the same as the system clock, but it has the inverted phase. When it is clear, the ADC0/1_ODD_PS and the ADC0/1_CLK_PS fields select the clock divide factor by which the system clock will be divided as shown in [Table 30-114](#). The ADC clock frequency is calculated as below and must not exceed ADC_MAX_CLOCK. This is also the maximum frequency of system clock when the ADC0/1_CLK_SEL is asserted.

$$\text{ADCClockFrequency} = \frac{\text{SystemClockFrequency(MHz)}}{\text{SystemClockDivideFactor}}$$

[Figure 30-138](#) depicts how the ADC clocks for ADC0 and ADC1 are generated.

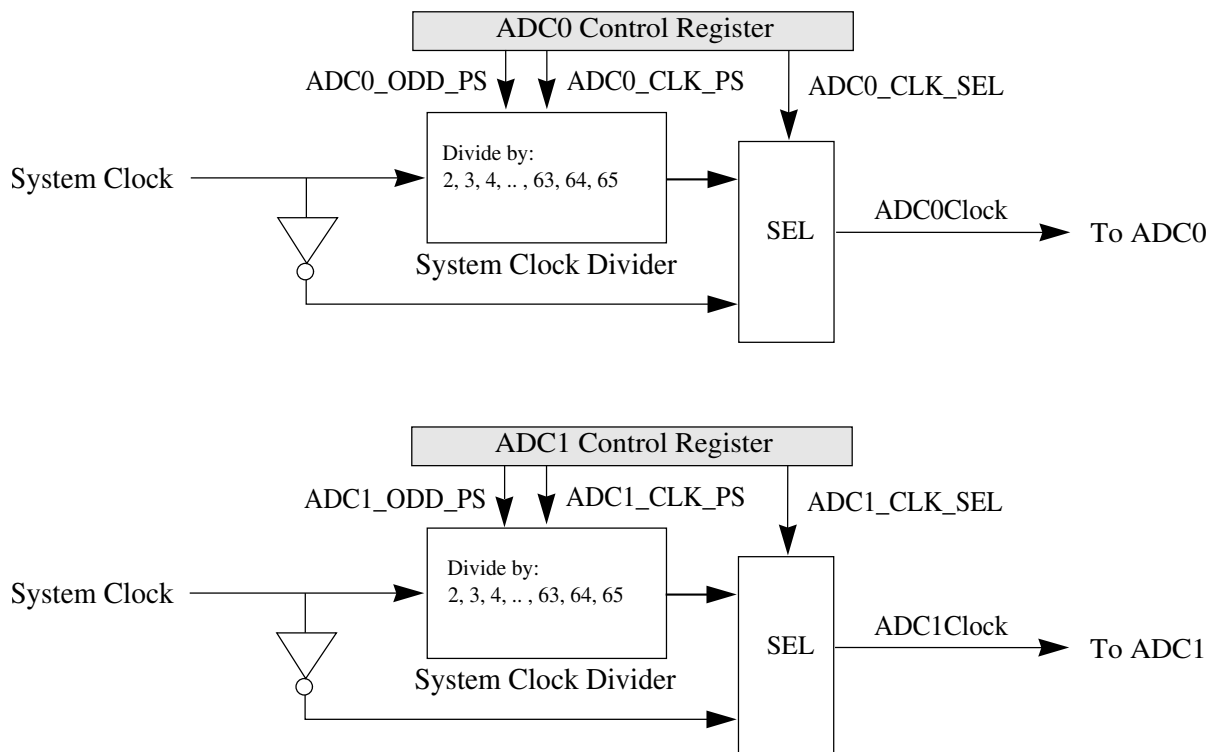


Figure 30-138. ADC0/1 clock generation

The ADC conversion speed (in K samples per second: Ksps) is calculated by the following formula. The *number of sampling cycles* is determined by the LST bits in the command message—see [Conversion command format for the standard configuration](#)—

and can take one of the following values: 2, 8, 64, or 128 ADC clock cycles. The *number of AD conversion cycles* is 13 for differential conversions and 14 for single-ended 12-bit resolution and unitary input gain conversions. The maximum conversion speed is achieved when the ADC Clock frequency is set to its maximum and the number of sampling cycles set to its minimum (2 cycles). The maximum conversion speed is ADC_MAX_CONV_RATE.

$$\text{ADCConversionSpeed} = \frac{\text{ADCClockFrequency(MHz)}}{(\text{NumberOfSamplingCycles} + \text{NumberOfADCConversionCycles})}$$

Table 30-137 shows an example of how the ADC0/1_CLK_PS can be set when using a 120 MHz system clock and the corresponding conversion speeds for all possible ADC clock frequencies. The table also shows that according to the system clock frequency, certain clock divide factors are invalid (2, 4, 6, 8 clock divide factors in the example) because their use would result in a ADC clock frequency higher than the maximum one supported by the ADC. The ADC clock frequency must not exceed ADC_MAX_CLOCK.

Table 30-137. ADC Clock configuration example (system clock frequency=120 MHz)

ADC0/1_CLK_PS [0:4]	ADC0/1_ODD_PS	System clock divide factor	ADC clock (system clock = 120 MHz)	Differential conversion speed with default sampling time (2 cycles)	Single-ended conversion speed with default sampling time (2 cycles)
0b00000	0	2	N/A	N/A	N/A
	1	3	N/A	N/A	N/A
0b00001	0	4	N/A	N/A	N/A
	1	5	N/A	N/A	N/A
0b00010	0	6	N/A	N/A	N/A
	1	7	N/A	N/A	N/A
0b00011	0	8	15.0 MHz	1.0 Msps	938 Ksps
	1	9	13.3 MHz	889 Ksps	833 Ksps
0b00100	0	10	12.0 MHz	800 Ksps	750 Ksps
	1	11	10.9 MHz	727 Ksps	682 Ksps
0b00101	0	12	10.0 MHz	667 Ksps	625 Ksps
	1	13	9.23 MHz	615 Ksps	577 Ksps
0b00110	0	14	8.57 MHz	571 Ksps	536 Ksps
	1	15	8.0 MHz	533 Ksps	500 Ksps
0b00111	0	16	7.5 MHz	500 Ksps	469 Ksps
	1	17	7.06 MHz	471 Ksps	441 Ksps
0b01000	0	18	6.67 MHz	444 Ksps	417 Ksps
	1	19	6.32 MHz	421 Ksps	395 Ksps
0b01001	0	20	6.0 MHz	400 Ksps	375 Ksps
	1	21	5.71 MHz	381 Ksps	357 Ksps

Table continues on the next page...

**Table 30-137. ADC Clock configuration example (system clock frequency=120 MHz)
(continued)**

ADC0/1_CLK_PS [0:4]	ADC0/1_ODD_PS	System clock divide factor	ADC clock (system clock = 120 MHz)	Differential conversion speed with default sampling time (2 cycles)	Single-ended conversion speed with default sampling time (2 cycles)
0b01010	0	22	5.45 MHz	364 Ksps	341 Ksps
	1	23	5.22 MHz	348 Ksps	326 Ksps
0b01011	0	24	5.0 MHz	333 Ksps	313 Ksps
	1	25	4.80 MHz	320 Ksps	300 Ksps
0b01100	0	26	4.62 MHz	308 Ksps	288 Ksps
	1	27	4.44 MHz	296 Ksps	278 Ksps
0b01101	0	28	4.29 MHz	286 Ksps	268 Ksps
	1	29	4.14 MHz	276 Ksps	259 Ksps
0b01110	0	30	4.0 MHz	267 Ksps	250 Ksps
	1	31	3.87 MHz	258 Ksps	242 Ksps
0b01111	0	32	3.75 MHz	250 Ksps	234 Ksps
	1	33	3.64 MHz	242 Ksps	227 Ksps
0b10000	0	34	3.53 MHz	235 Ksps	221 Ksps
	1	35	3.43 MHz	229 Ksps	214 Ksps
0b10001	0	36	3.33 MHz	222 Ksps	208 Ksps
	1	37	3.24 MHz	216 Ksps	203 Ksps
0b10010	0	38	3.16 MHz	211 Ksps	198 Ksps
	1	39	3.08 MHz	205 Ksps	192 Ksps
0b10011	0	40	3.0 MHz	200 Ksps	188 Ksps
	1	41	2.93 MHz	195 Ksps	183 Ksps
0b10100	0	42	2.86 MHz	190 Ksps	179 Ksps
	1	43	2.79 MHz	186 Ksps	174 Ksps
0b10101	0	44	2.73 MHz	182 Ksps	170 Ksps
	1	45	2.67 MHz	178 Ksps	167 Ksps
0b10110	0	46	2.61 MHz	174 Ksps	163 Ksps
	1	47	2.55 MHz	170 Ksps	160 Ksps
0b10111	0	48	2.5 MHz	167 Ksps	156 Ksps
	1	49	2.45 MHz	163 Ksps	153 Ksps
0b11000	0	50	2.4 MHz	160 Ksps	150 Ksps
	1	51	2.35 MHz	157 Ksps	147 Ksps
0b11001	0	52	2.31 MHz	154 Ksps	144 Ksps
	1	53	2.26 MHz	151 Ksps	142 Ksps
0b11010	0	54	2.22 MHz	148 Ksps	139 Ksps
	1	55	2.18 MHz	145 Ksps	136 Ksps
0b11011	0	56	2.14 MHz	143 Ksps	134 Ksps
	1	57	2.11 MHz	140 Ksps	132 Ksps

Table continues on the next page...

**Table 30-137. ADC Clock configuration example (system clock frequency=120 MHz)
(continued)**

ADC0/1_CLK_PS [0:4]	ADC0/1_ODD_PS	System clock divide factor	ADC clock (system clock = 120 MHz)	Differential conversion speed with default sampling time (2 cycles)	Single-ended conversion speed with default sampling time (2 cycles)
0b11100	0	58	2.07 MHz	138 Ksps	129 Ksps
	1	59	2.03 MHz	136 Ksps	127 Ksps
0b11101	0	60	2.0 MHz	133 Ksps	125 Ksps
	1	61	1.97 MHz	131 Ksps	123 Ksps
0b11110	0	62	1.94 MHz	129 Ksps	121 Ksps
	1	63	1.90 MHz	127 Ksps	119 Ksps
0b11111	0	64	1.88 MHz	125 Ksps	117 Ksps
	1	65	1.85 MHz	123 Ksps	115 Ksps

30.6.6.3 Time stamp feature

The on-chip ADCs can provide a time stamp for the conversions they execute. A time stamp is the value of the time base counter latched when the EQADC detects the end of the analog input voltage sampling. A time stamp for a conversion command is requested by setting the TSR bit in the corresponding command. When TSR is negated, that is a time stamp is not requested, the ADC returns a single result message containing the conversion result. When TSR is asserted, that is a time stamp is requested, the ADC returns two result messages; one containing the conversion result, and afterwards another containing the time stamp for that conversion. The result messages are sent in this order to the RFIFOs and both messages are sent to the same RFIFO was specified in the MESSAGE_TAG field of the executed conversion command.

The time stamp can be provided by an external source using the STAC bus (more details in [STAC bus client sub-module \(STACC\)](#)) or by the internal time base counter. The selection between the two sources is done by field ADC0/1_TBSEL in the ADC0/1_CR register or by field ATBSEL in registers ADC_ACR1-14. See [Table 30-113](#) and the ATBSEL field description in [Alternate Configuration 1-14 Control Registers \(ADC_ACR1-14\)](#) for selection details.

The time base counter is a 16-bit up counter that wraps after reaching 0xFFFF. It is disabled after reset and it is enabled according to the setting of TBC_CLK_PS field in [ADC Time Stamp Control Register \(ADC_TSCR\)](#). TBC_CLK_PS defines if the counter is enabled or disabled, and, if enabled, at what frequency it is incremented. The time stamps are returned regardless of whether the time base counter is enabled or disabled. The time base counter can be reset by writing 0x0000 to [ADC Time Base Counter Registers \(ADC_TBCR\)](#) with a write configuration command.

30.6.6.3.1 STAC bus client sub-module (STACC)

The STAC sub-block STACC extracts the external time base from the STAC bus to the EQADC. Figure 30-139 provides a block diagram for the STACC sub-module that contains actually two slot selectors.

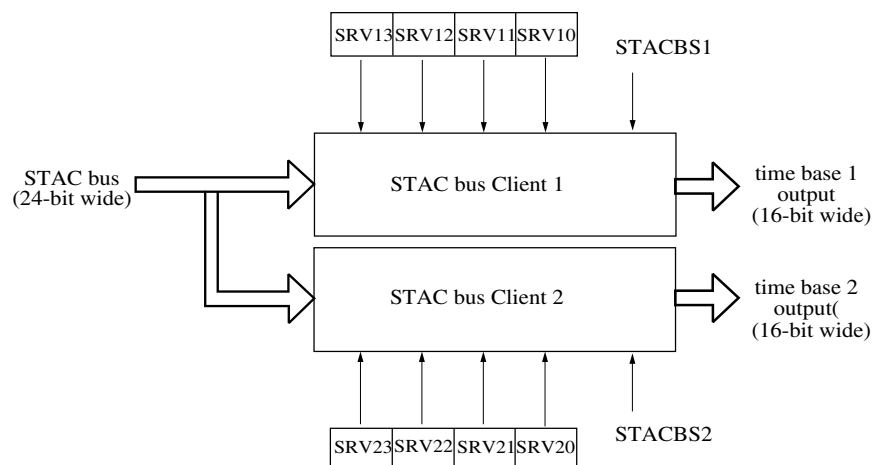
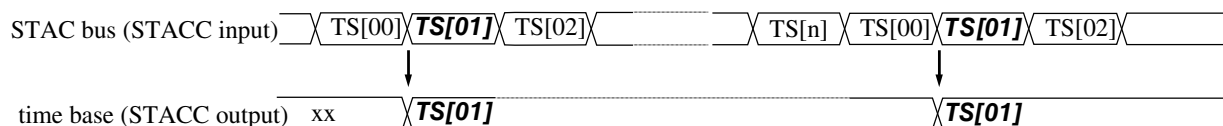


Figure 30-139. STAC bus client block diagram

The STAC bus data stream is composed by several identified time slots. Each time slot contains a time value from a specified source. The bits SRV1/2[0:3] in register EQADC_STACCR are used to select the desired time slots of the STAC bus to be used internally by the EQADC. Figure 30-140 shows a timing diagram for the STACC.



- NOTES: 1. Maximum of 16 Time Slots (TS[n])
2. In this case, SRV bits were set to capture TS[01]

Figure 30-140. Timing diagram for the STAC bus and STACC output

Every time the selected time slot change, the STACC outputs are updated. As this time base is an external data to the EQADC, this output is not affected by the stop or by the debug internal state.

After the slot selection is done and the timebase data is extracted, the STACBS1/2 bits select 16 bits from the original 24-bit timebase data. These selected bits are the timebase to be used internal to the EQADC.

30.6.6.4 ADC pre-gain feature

Each ADC can be configured to have a selectable input gain as defined in [Alternate Configuration 1-14 Control Registers \(ADC_ACR1-14\)](#). This means the input signal is sampled and the result is amplified by factor 2, or 4 before the conversion phase. In present implementation of this feature, the conversion is 1 or 2 ADC clock cycles longer for gain 2 or gain 4, respectively.

30.6.6.5 ADC resolution selection feature

The ADCs conversion resolutions can be 8 bits, 10 bits or 12 bits as described in [Alternate Configuration 1-14 Control Registers \(ADC_ACR1-14\)](#). For conversions at a resolution less than 12, the ADC is executing less operations and the conversion time is smaller. In this ADC, it is verified that there is 1 ADC clock cycle for each bit of resolution. Therefore, for the same ADC clock frequency, the ADC sample frequency is higher for lower resolutions.

When a conversion is undertaken at a resolution less than 12, the result is presented by the ADC in right justified format in the 12-bit input bus e.g.: 0000xxxxxxx for 8 bits and 00xxxxxxxxxx for 10 bits. The EQADC inverts the result to left justified format i.e.: xxxxxxxx0000 for 8 bits and xxxxxxxxxx00 for 10 bits. This is because the same calibration coefficients in the MAC can then be used. The left shift operation is done just after the conversion result enters the EQADC, in the Resolution Adjustment block prior to the MAC, as illustrated in [Figure 30-143](#).

30.6.6.6 ADC calibration feature

30.6.6.6.1 Overview

There are three sets of calibration coefficients for each ADC. Each set is composed by a gain factor and an offset factor: GCCn/OCCn, ALTGCCn1/ALTGCCn1, and ALTGCCn2/ALTGCCn2, where n is the ADC number 0 or 1. The pair GCCn/OCCn is selected when it is used the normal configuration or the alternate configurations 3 to 8. The pair ALTGCCn1/ALTGCCn1 is used only when the alternate configuration 1 is selected. And the pair ALTGCCn2/ALTGCCn2 is for the alternate configuration 2. The description below is for a generic pair of gain/offset GCC/OCC.

The EQADC provides a calibration scheme to remove the effects of gain and offset errors from the results generated by the on-chip ADCs. Only results generated by the on-chip ADCs are calibrated. The main component of calibration hardware is a Multiply-and-Accumulate (MAC) unit, one per on-chip ADC, that is used to calculate the following transfer function which relates a calibrated result to a raw, uncalibrated one.

$$\text{CAL\_RES} = \text{GCC} * \text{RAW\_RES} + \text{OCC} + 2;$$

where:

- CAL_RES is the calibrated result corresponding the input voltage V_i .
- GCC is the gain calibration constant.
- RAW_RES is the raw, uncalibrated result with resolution adjustment corresponding to an specific input voltage V_i .
- OCC is the offset calibration constant.
- The addition of two reduces the maximum quantization error of the ADC. See [Quantization error reduction during calibration](#).

Calibration constants GCC and OCC are determined by taking two samples of known reference voltages and using these samples to calculate the values for the constants. For details and an example about how to calculate the calibration constants and use them in result calibration see [ADC Result Calibration](#). Once calculated, GCC is stored in [ADC0/1 Gain Calibration Constant Registers \(ADC0_GCCR and ADC1_GCCR\)](#) and OCC in [ADC0/1 Offset Calibration Constant Registers \(ADC0_OCCR and ADC1_OCCR\)](#), from where their values are fed to the MAC unit. The alternate gain values are stored in [ADC0/1 Alternate Gain Registers \(ADC0_AGR1-2 and ADC1_AGR1-2\)](#) and the alternate offset values in [ADC0/1 Alternate Offset Register \(ADC0_AOR1-2 and ADC1_AOR1-2\)](#). Since the analog characteristics of each on-chip ADCs differs, each ADC has an independent pair of calibration constants.

A conversion result is calibrated according to the status of CAL bit in the command that initiated the conversion. If the CAL bit is asserted, the EQADC will automatically calculate the calibrated result before sending the result to the appropriate RFIFO or companion module. If the CAL bit is negated, the result is not calibrated, it bypasses the calibration hardware, and is directly sent to the appropriate RFIFO or companion module.

30.6.6.6.2 MAC unit and operand data format

The MAC unit diagram is shown in [Figure 30-141](#). Each on-chip ADC has a separate MAC unit to fine-tune its conversion results. The description below considers the general calibration constant registers but it is the same for the alternate calibration constants.

The OCC0/1 operand is a 14-bit signed value stored in [ADC0/1 Offset Calibration Constant Registers \(ADC0_OCCR and ADC1_OCCR\)](#)." The RAW_RES operand is the raw uncalibrated result, and it is the direct output from the on-chip ADCs but passing through the resolution adjustment block. The GCC0/1 operand is a 15-bit fixed point unsigned value stored in [ADC0/1 Gain Calibration Constant Registers \(ADC0_GCCR and ADC1_GCCR\)](#)." The GCC is expressed in the *GCC_INT.GCC_FRAC* binary format. The integer part of the GCC (GCC_INT=GCC[1]) contains a single binary digit while its fractional part (GCC_FRAC=GCC[2:15]) contains 14 bits - see [Figure 30-142](#). The gain constant equivalent decimal value ranges from 0 to 1.999938..., as shown in [Table 30-139](#). Two is always added to the MAC output - see [Quantization error reduction during calibration](#). CAL_RES output is the calibrated result, and it is a 14-bit unsigned value. CAL_RES is truncated to 0x3FFF, in case of a overflow, and to 0x0000, in case of an underflow.

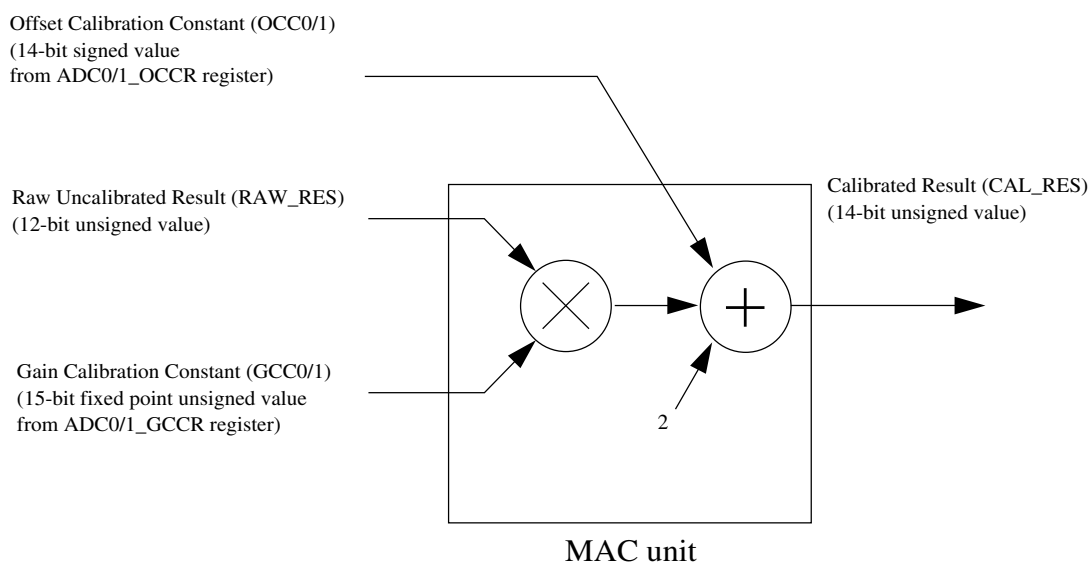


Figure 30-141. MAC unit diagram

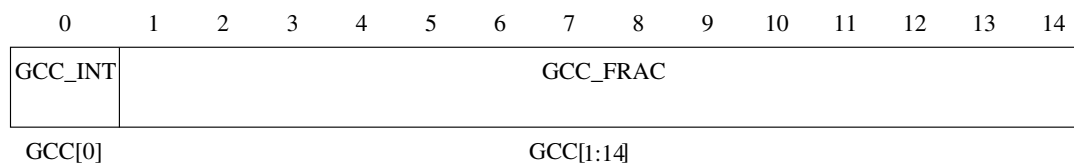


Figure 30-142. Gain calibration constant format

Table 30-138. Gain calibration constant format — field descriptions

Field	Description
GCC_INT	Integer part of the gain calibration constant for ADC0/1GCC_INT is the integer part of the gain calibration constant for ADC0/1.

Table continues on the next page...

Table 30-138. Gain calibration constant format — field descriptions (continued)

Field	Description
GCC_FRAC[1:14]	Fractional part of the gain calibration constant for ADC0/1. See Table 30-139 . GCC_FRAC is the fractional part of the gain calibration constant for ADC0/1. GCC_FRAC can express decimal values ranging from 0 to 0.999938...

Table 30-139. Binary and decimal representations of the gain constant

Gain constant (GCC_INT.GCC_FRAC binary format)	Corresponding decimal value
0.0000_0000_0000_00	0
...	...
0.1000_0000_0000_00	0.5
...	...
0.1111_1111_1111_11	0.999938...
1.0000_0000_0000_00	1
...	...
1.1100_0000_0000_00	1.75
...	...
1.1111_1111_1111_11	1.999938...

30.6.6.7 ADC control logic overview and command execution

[Figure 30-143](#) shows the basic logic blocks involved in the ADC Control and how they interact. CFIFOs/RFIFOs interact with CBuffers/*Abort Cont/Result Message Return Logic* through the *FIFO Control Unit*. The EB and BN bits in the Command Message uniquely identify the CBuffer to which a command should be sent. The *FIFO Control Unit* decodes these bits and sends the ADC command to the proper CBuffer. Other blocks of logic are the *Resolution Adjustment*, *Result Format and Calibration Sub-Block*, the *Time Stamp Logic*, and the *MUX Control Logic*.

The *Resolution Adjustment Sub-Block* receives the 12-bit data bus directly from the ADC and changes the received conversion results from right aligned format of ADC to the left aligned format depending on the selected resolution of the conversion. This operation helps the calibration processing to use the calibration coefficients always with the same format.

The *Result Format and Calibration Sub-Block* formats the returning data into Result Messages and sends them to the RFIFOs ⁶. The returning data can be data read from an ADC register, a conversion result, or a time stamp. The formatting and calibration of conversion results also take place inside this sub-block.

The *Time Stamp Logic* latches the value of the time base counter or the STAC bus time base when detecting the end of the analog input voltage sampling, and sends it to the *Result Format and Calibration Sub-Block* as time stamp information.

The *MUX Control Logic* generates the proper MUX control signals and, when the ADC0/1_EMUX bits are asserted, the MA signals based on the channel numbers extracted from the ADC Command.

When the on-chip ADC abort feature is not enabled, ADC Commands are stored in the CBuffers as they come and they are executed in the first-in-first-out basis. After the execution of a command in ENTRY1 finishes all commands are shifted one entry. After the shift, ENTRY0 is always empty and ready to receive a new command. Execution of configuration commands only start when they reach ENTRY1. Consecutive conversion commands are pipelined and their execution can start while in ENTRY0. This is explained below.

AD conversion accuracy can be affected by the settling time of the input channel multiplexers. Some time is required for the channel multiplexers internal capacitances to settle after the channel number is changed. If the time prior to sampling is not long enough to absorb this settling, then the settling time will take from ADC sampling time which may result in inaccurate sampling and ultimately compromise conversion result accuracy - see [Figure 30-144](#) (a). This could be avoided by switching the multiplexers in preparation for the next command's sampling during the AD conversion phase of the current command as showed in [Figure 30-144](#) (b). In EQADC, this is done in the following way; when a conversion command is in buffer ENTRY1 and another conversion command is identified in ENTRY0, then the channel number of ENTRY0 is sent to the *MUX Control Logic* some cycles before the sampling phase of the command in ENTRY0 starts. In this way, sampling for the next command can promptly start after the current conversion finishes because the internal capacitance of the multiplexers will be settled by that time, allowing for more accurate sampling. This is specially important for applications that require high conversion speeds, that is with the ADC running at maximum clock frequency and with the analog input voltage sampling time set to a minimum (2 ADC clock cycles), when the short sampling time does not allow the multiplexers to completely settle. The second advantage of pipelining conversion

6. The result messages may also be routed to an on-chip companion module via the side interface, and then fed back to the RFIFOs.

commands is to provide precise conversion intervals, which means the time intervals between two consecutive conversions are the same. This is important for any digital signal process application.

When the on-chip ADC abort feature is enabled, ADC Commands from CFIFO0 should be considered immediately, even stopping the execution of some command that is already in ENTRY1. When the abort request is sent to the ADC, the already stored commands in the CBuffers are copied in a temporary set of registers. The first ADC command from CFIFO0 is sent after the abort acknowledge indication from ADC. The process is the same as usual until the transfer of the last command from CFIFO0. Then the temporarily stored commands that were postponed by the abortion are recovered and they are pipelined for execution. After the last command from this temporary memory is transferred, the next commands are pipelined from the CFIFOs.

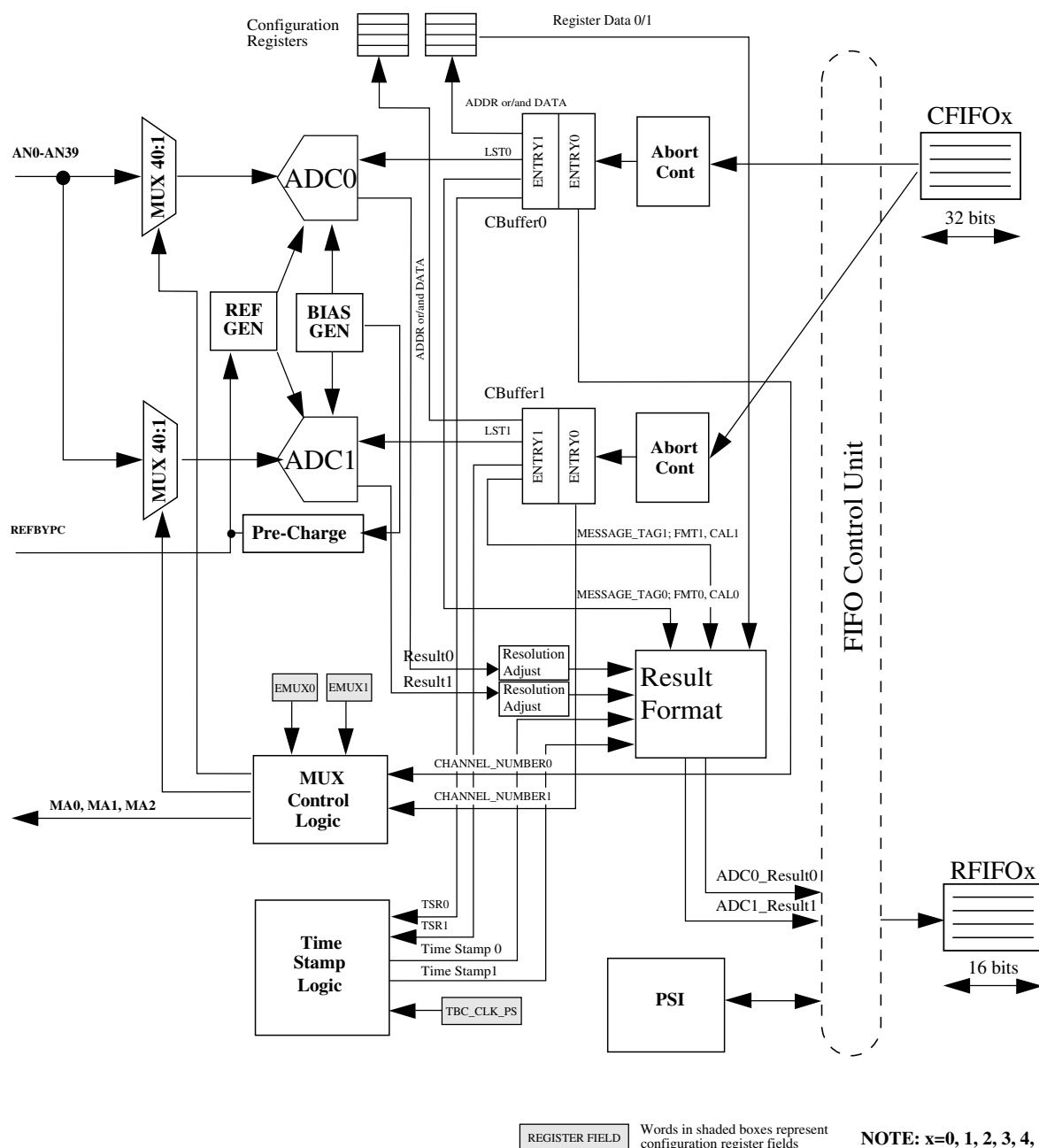
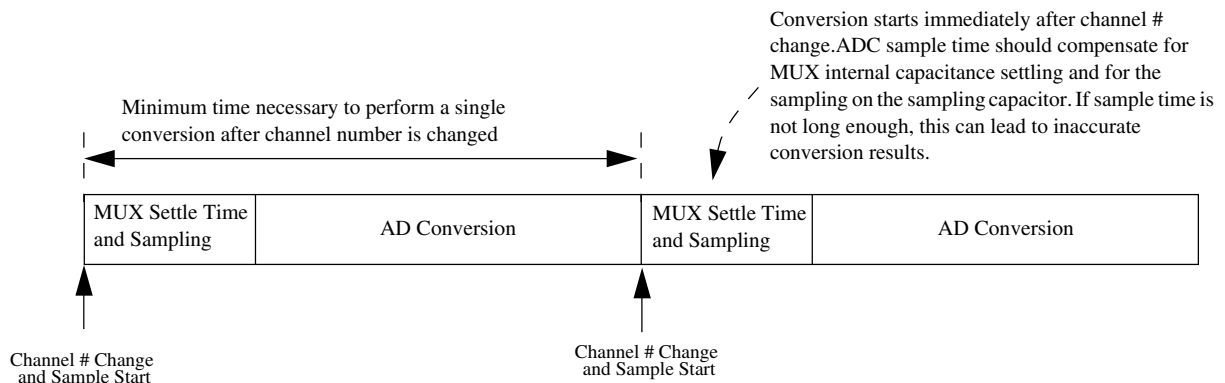
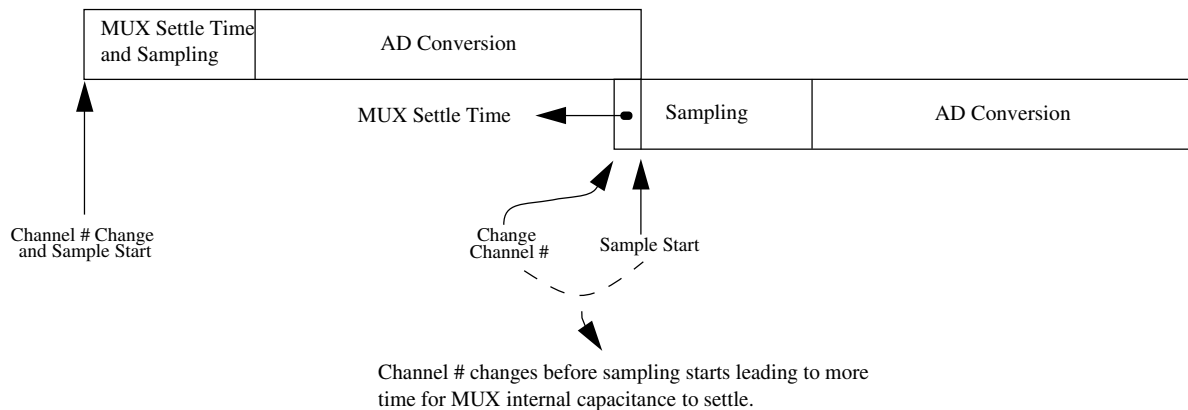


Figure 30-143. On-chip ADC control scheme



(a) Command Execution Sequence for Two Non-Overlapped Commands



(b) Command Execution Sequence for Two Overlapped Commands

Figure 30-144. Overlapping consecutive conversion commands

30.6.7 Internal/external multiplexing

30.6.7.1 Channel assignment

The internal analog multiplexers select one of the 40 analog input pins for conversion, based on the `CHANNEL_NUMBER` field of a Command Message. The analog input pin channel number assignments and the pin definitions vary depending on how the `ADC0/1_EMUX` are configured. Allowed combinations of `ADC0/1_EMUX` bits are shown in [Table 30-140](#) together with references to tables indicating how `CHANNEL_NUMBER` field of each conversion command must be set to avoid channel selection conflicts.

During differential conversions the analog multiplexer passes differential signals to both the positive and negative terminals of the ADC. The differential conversions can only be initiated on four channels: DAN0, DAN1, DAN2, and DAN3. See chip-specific section for the channel numbers used to select differential conversions.

Table 30-140. ADC0/1_EMUX bits combinations

ADC0_EMUX	ADC1_EMUX	CHANNEL_NUMBER should be set as in	
		ADC0	ADC1
0	0	See chip-specific section for details	See chip-specific section for details
0	1	See chip-specific section for details	See chip-specific section for details
1	0	See chip-specific section for details	See chip-specific section for details
1	1	Reserved ¹	

1. ADC0_EMUX and ADC1_EMUX must not be asserted at the same time

See the chip-specific EQADC information for channel number assignments for the non-multiplexed mode. The two on-chip ADCs can access the same analog input pins but simultaneous conversions are not allowed. Also, when one ADC is performing a differential conversion on a pair of pins, the other ADC must not access either of these two pins as single-ended channels. The 43 single-ended channels and 4 differential pairs are shared between the two ADCs.

See the chip-specific EQADC information for the channel number assignments for multiplexed mode. The two on-chip ADCs can access the same analog input pins but simultaneous conversions are not allowed. Also, when one ADC is performing a differential conversion on a pair of pins, the other ADC must not access either of these two pins as single-ended channels. The ADC with the ADC0/1_EMUX bit asserted can access 4 differential pairs, 39 single-ended, and, at most, 64 externally multiplexed channels. See [External multiplexing](#) for a detailed explanation about how external multiplexing can be achieved.

30.6.7.2 External multiplexing

The EQADC can use from one to eight external multiplexer chips to expand the number of analog signals that may be converted. Up to 64 analog channels can be converted through external multiplexer selection. The externally multiplexed channels are automatically selected by the CHANNEL_NUMBER field of a Command Message, in the same way done with internally multiplexed channels. The software selects the external multiplexed mode by setting the ADC0/1_EMUX bit in either ADC0_CR or ADC1_CR depending on which ADC will perform the conversion. See the chip-specific EQADC information for the channel number assignments for the multiplexed mode.

There are 4 differential pairs, 39 single-ended, and, at most, 64 externally multiplexed channels which can be selected. Only one ADC can have its ADC0/1_EMUX bit asserted at a time.

Figure 30-145 shows the maximum configuration of eight external multiplexer chips connected to the EQADC. The external multiplexer chip selects one of eight analog inputs and connects it to a single analog output, which is fed to a specific input of the EQADC. The EQADC provides three multiplexed address signals, MA0, MA1, and MA2, to select one of eight inputs. These three multiplexed address signals are connected to all eight external multiplexer chips. The analog output of the eight multiplex chips are each connected to eight separate EQADC inputs, ANR, ANS, ANT, ANU, ANW, ANX, ANY, and ANZ. The MA pins correspond to the three least significant bits of the channel number that selects ANR, ANS, ANT, ANU, ANW, ANX, ANY, and ANZ with MA0 being the most significant bit - See Table 30-141. In that table, '0' means pin is driven LOW and '1' that pin is driven HIGH.

Table 30-141. Encoding of MA pins

Channel number selecting ANR, ANS, ANT, ANU, ANW, ANX, ANY, ANZ (decimal)								MA0	MA1	MA2
ANR	ANS	ANT	ANU	ANW	ANX	ANY	ANZ			
224	232	240	248	64	72	80	88	0	0	0
225	233	241	249	65	73	81	89	0	0	1
226	234	242	250	66	74	82	90	0	1	0
227	235	243	251	67	75	83	91	0	1	1
228	236	244	252	68	76	84	92	1	0	0
229	237	245	253	69	77	85	93	1	0	1
230	238	246	254	70	78	86	94	1	1	0
231	239	247	255	71	79	87	95	1	1	1

When the external multiplexed mode is selected for either ADC, the EQADC automatically creates the MA output signals from CHANNEL_NUMBER field of a Command Message. The EQADC also converts the proper input channel (ANR, ANS, ANT, ANU, ANW, ANX, ANY, and ANZ) by interpreting the CHANNEL_NUMBER field. As a result, up to 64 externally multiplexed channels appear to the conversion queues as directly connected signals.

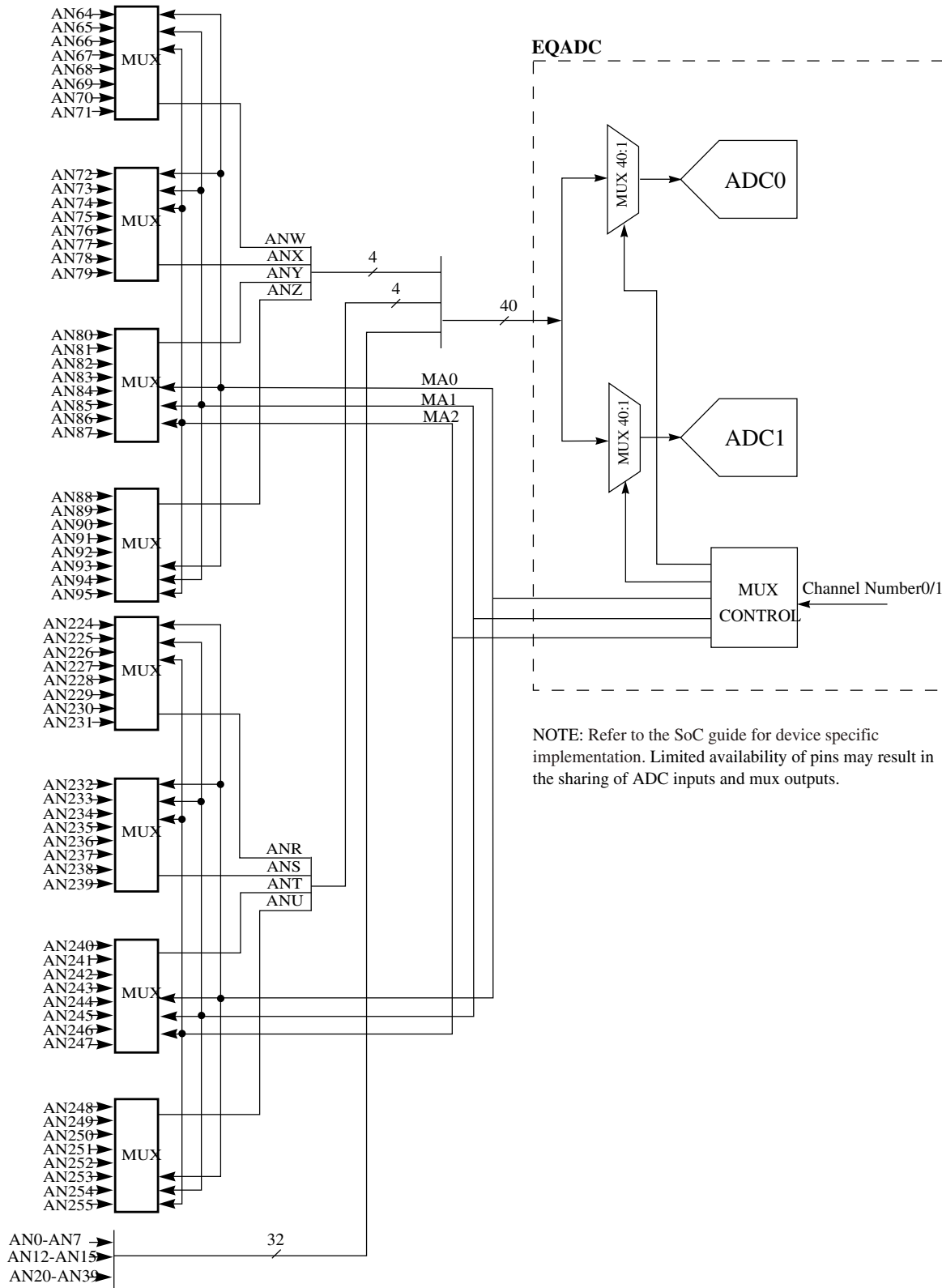


Figure 30-145. Example of external multiplexing

30.6.8 EQADC DMA/interrupt request

Table 30-142 lists methods to generate interrupt requests in the EQADC queuing control and triggering control. The DMA/interrupt request select bits and the DMA/interrupt enable bits are described in [EQADC Interrupt and DMA Control Register 0 \(EQADC_IDCR0\)](#) and the interrupt flag bits are described in [EQADC FIFO and Interrupt Status Register \(EQADC_FISR_n\)](#). Figure 30-146 depicts all interrupts and DMA requests generated by the EQADC. For details, see [EQADC FIFO and Interrupt Status Register \(EQADC_FISR_n\)](#) and [EQADC Interrupt and DMA Control Register 0 \(EQADC_IDCR0\)](#).

Table 30-142. EQADC FIFO interrupt summary

Interrupt	Condition	Clearing mechanism
Non Coherency Interrupt	NClEx = 1 NCFx = 1	Clear NCFx bit by writing a "1" to the bit.
Result FIFO Overflow Interrupt ¹	RFOlEx = 1 RFOF _x = 1	Clear RFOF _x bit by writing a "1" to the bit.
Command FIFO Underflow Interrupt ¹	CFUIEx = 1 CFUF _x = 1	Clear CFUF _x bit by writing a "1" to the bit.
Result FIFO Drain Interrupt	RFDE _x = 1 RFDS _x = 0 RFDF _x = 1	Clear RFDF _x bit by writing a "1" to the bit.
Command FIFO Fill Interrupt	CFFEx = 1 CFFS _x = 0 CFFF _x = 1	Clear CFFF _x bit by writing a "1" to the bit.
End of Queue Interrupt	EOQlEx = 1 EOQF _x = 1	Clear EOQF _x bit by writing a "1" to the bit.
Pause Interrupt	PIEx = 1 PF _x = 1	Clear PF _x bit by writing a "1" to the bit.
Trigger Overrun Interrupt ¹	TORlEx = 1 TORF _x = 1	Clear TORF _x bit by writing a "1" to the bit.

1. Apart from generating an independent interrupt request for when a RFIFO Overflow Interrupt, a CFIFO Underflow Interrupt, and a CFIFO Trigger Overrun Interrupt occurs, the EQADC also provides a combined interrupt request at which these requests from ALL CFIFOs are ORed. Refer to [Figure 30-146](#) for details.

Table 30-143 describes a list of methods to generate DMA requests in the EQADC. For details, see [EQADC FIFO and Interrupt Status Register \(EQADC_FISR_n\)](#) and [EQADC Interrupt and DMA Control Register 0 \(EQADC_IDCR0\)](#).

Table 30-143. EQADC FIFO DMA Summary

DMA Request	Condition	Clearing Mechanism
Result FIFO Drain DMA ¹ Request	RFDEx = 1 RFDSx = 1 RFDFx = 1	The EQADC automatically clears the RFDFx when RFIFOx becomes empty. Writing "1" to the RFDFx bit is not allowed.
Command FIFO Fill DMA Request	CFFEx = 1 CFFSx = 1 CFFFx = 1	The EQADC automatically clears the CFFFx when CFIFOx becomes full. Writing "1" to the CFFFx bit is not allowed.

1. Additional readings from RFIFOx (not from DMA control) can clear the RFDFx bit and can lock the DMAx circuit to a locked state. The DMAx request remains ON, despite RFDFx is cleared.

On-chip ADC configuration and control

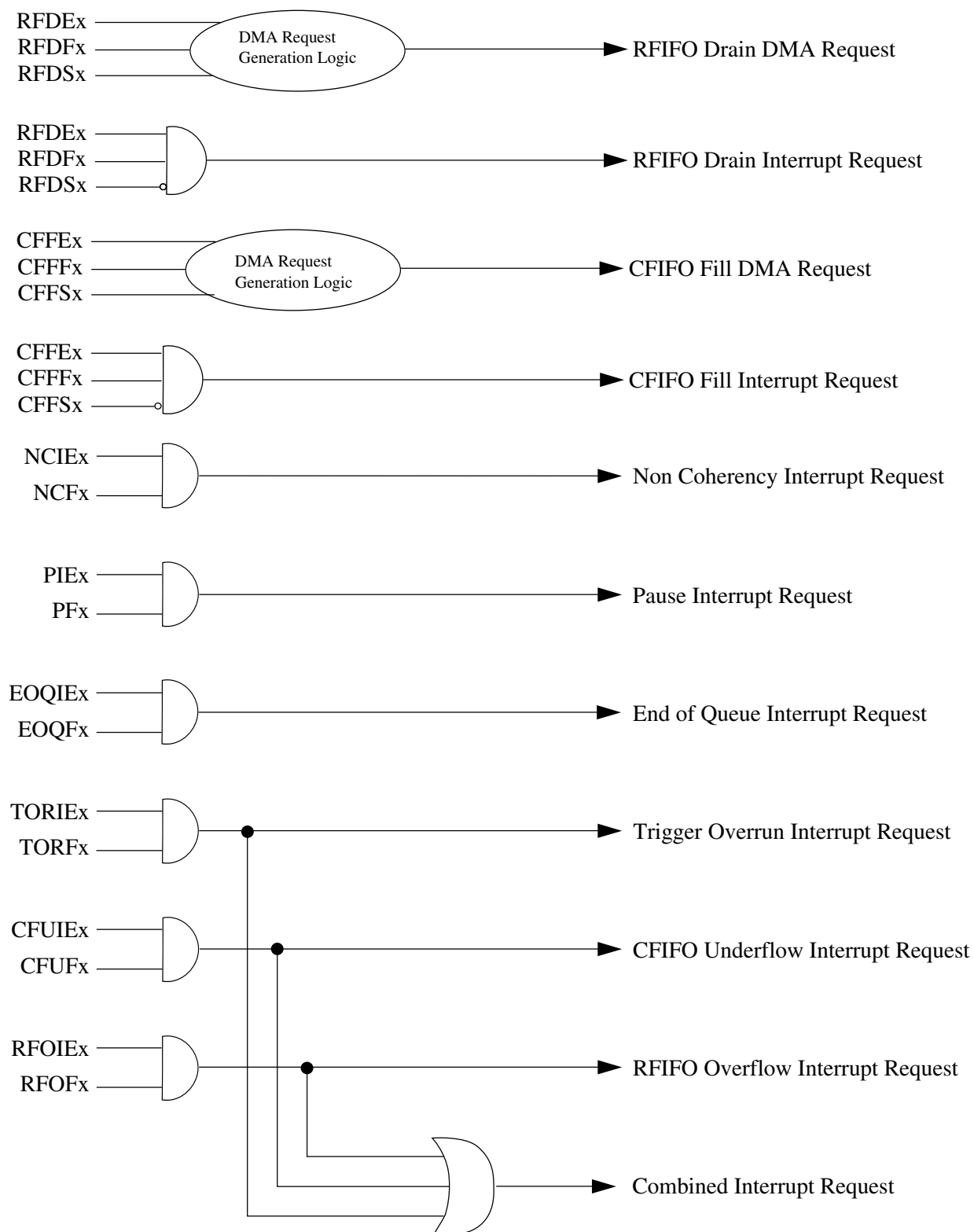


Figure 30-146. EQADC DMA and interrupt requests

30.6.9 EQADC Parallel Side Interface (PSI) sub-block

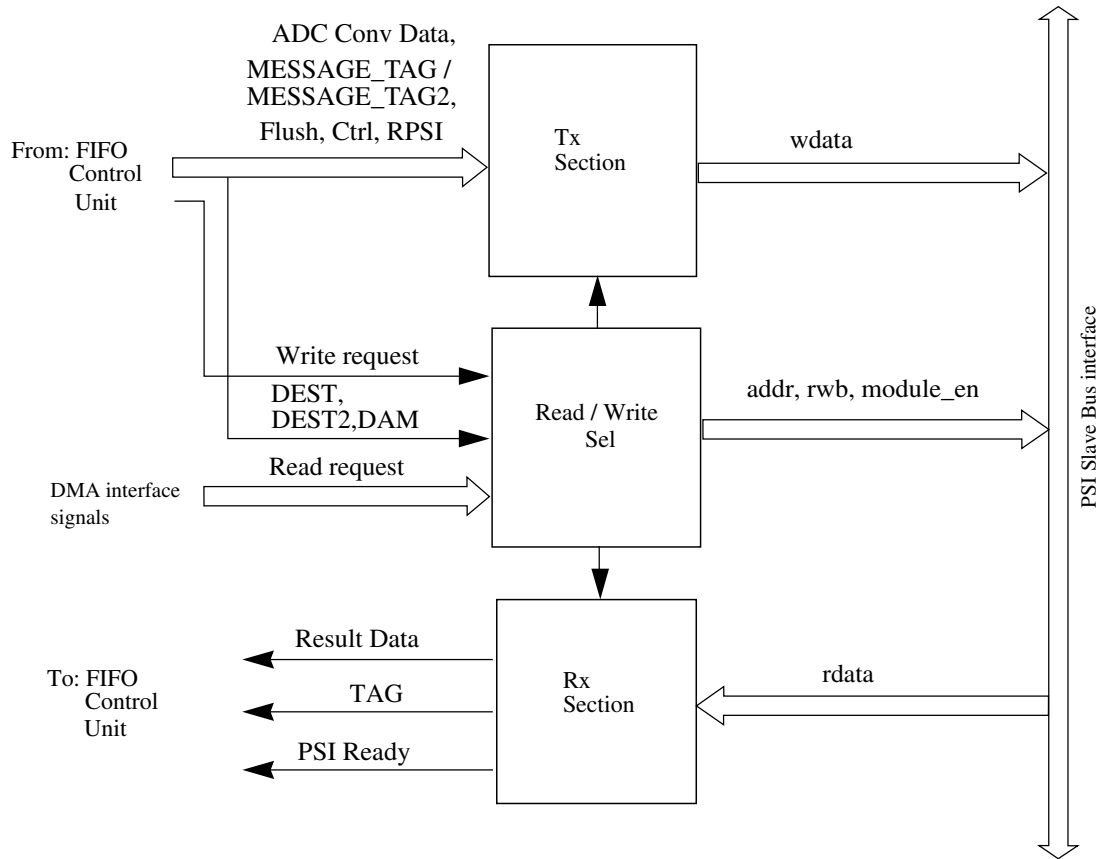


Figure 30-147. EQADC Parallel Side Interface block diagram

The EQADC PSI sub-block allows communication between the EQADC and the companion modules through a local slave bus that has the EQADC block as the master and the companion modules as the slaves. However, due to implementation constraints, a limited number of addresses is defined and a maximum of 15 different slave blocks to be enabled.

The conversion result goes through the Result Formatting and Calibration sub-block and the corresponding MESSAGE_TAG / MESSAGE_TAG2 and DEST/DEST2 fields are decoded to decide the destination of the conversion result data. When the DEST field is not zero as described in [Alternate Configuration 1-14 Control Registers \(ADC_ACR1-14\)](#), the MESSAGE_TAG bits, control bits, and the conversion result data are sent to the transmission section of the PSI sub-block. This set of data is processed and sent to the corresponding on-chip companion module.

The configuration bit DAM is described in [EQADC Module Configuration Register \(EQADC_MCR\)](#). The DEST2/MESSAGE_TAG2 fields are only used as defined in [Extended Alternate Configuration 1-14 Control Registers \(ADC_EACR1-14\)](#) when

DAM=1. DAM bit selects between two different "tap modes", each defining a different way to select the second destination. When DAM=0 the DEST field can set the transmission section to send the same result data / MESSAGE_TAG to two destinations. See [Table 30-121](#) in [Alternate Configuration 1-14 Control Registers \(ADC_ACR1-14\)](#). When DAM=1, the two destinations are selected as described in [Table 30-125](#).

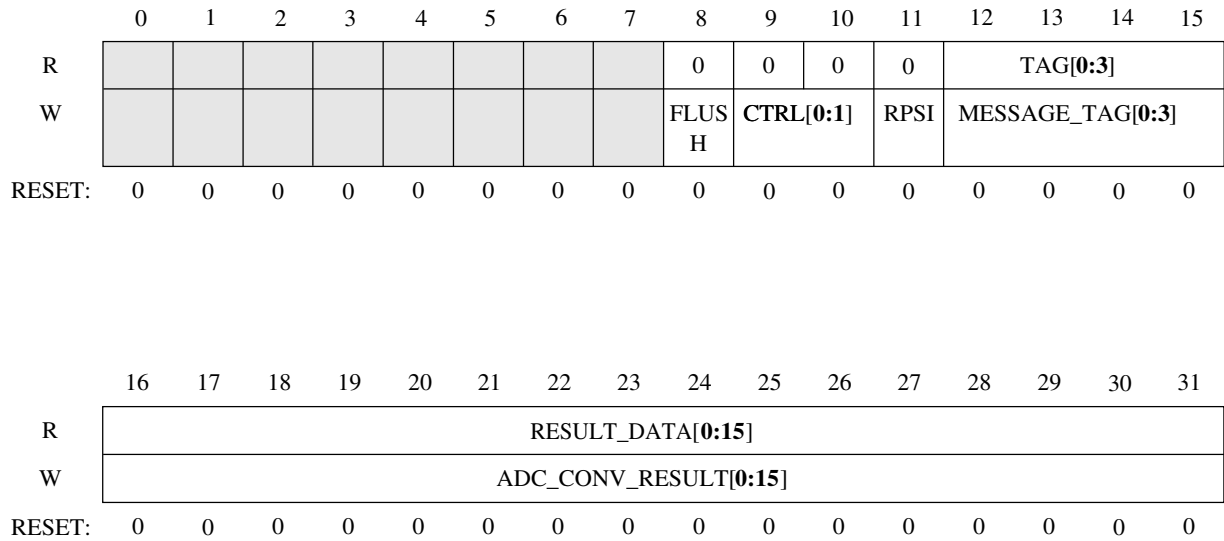
The companion module can also send back to EQADC the result of some processing in the received data. In this case, the receiver section of the PSI sub-block treats this request and receives the result data with the corresponding MESSAGE_TAG or TAG field. This TAG is used by the decoder to select which RFIFO will be written with the received companion module data.

A companion module with two PSIs can be connected to two EQADCs. For this case, there is the bit field RPSI described in [Alternate Configuration 1-14 Control Registers \(ADC_ACR1-14\)](#) and the bit field RPSI2 described in [Alternate Configuration 1-14 Control Registers \(ADC_ACR1-14\)](#) that control the routing of the companion module output result to any PSI.

30.6.9.1 Input / output signals description

The information content of the input and the output data buses from/to the companion modules are described in [Figure 30-148](#). The input data format is used in the Read cycle and corresponds to the data supplied by some companion module. The output data format is used in the Write cycle of this interface block and contains information / controls to some companion module.

Read (Input) / Write (Output) Data Buses Content



 = Unimplemented or Reserved

Figure 30-148. PSI input and output data buses content

Table 30-144. PSI input and output data buses content — field descriptions

Field	Description
FLUSH	Master block Flush request/control bit The FLUSH signal is used to request a Flush flow in the slave block. More details is presented in Conversion command format for alternate configurations .
	0 No flush request.
	1 Flush request
RPSI	Return Parallel Side Interface The RPSI bit is used in the companion module that has two PSI interfaces (PSI0 and PSI1) which can be connected to two eQADCs. It specifies in the companion module if the data received from one PSI should be transmitted to the same PSI connected to this eQADC or to the other one. This bit is defined by the bit field RPSI described in Alternate Configuration 1-14 Control Registers (ADC_ACR1-14) or by the bit field RPSI2 described in Alternate Configuration 1-14 Control Registers (ADC_ACR1-14) .
	0 Results from this sample must be transmitted back using the same PSI (eQADC).
	1 Results from this sample must be transmitted to the other PSI (eQADC)
CTRL	Control bits This field is used for the control of the companion module.
	00 INH_RET or PREFILL - This mode indicates that the companion module should not send back some RESULT_DATA corresponding to the accompanying ADC_CONV_RESULT data. When the slave module is the Decimation Filter, this control enables the PREFILL filter mode.

Table continues on the next page...

Table 30-144. PSI input and output data buses content — field descriptions (continued)

Field	Description
	01 CONVERSION RESULT - It indicates that the ADC_CONV_RESULT field should be treated as a valid sample data. This control mode is useful to Decimation Filter to put the filter em normal mode instead of prefill mode.
	10 TIME STAMP o REGISTER READ - A time stamp indicates that the ADC_CONV_RESULT field should follow a bypass flow in the companion module, returning back to the EQADC without any modification.
	11 Reserved
MESSAGE_TAG[0:3]	Message tag bits field This field indicates the RFIFO destination associated with the ADC_CONV_RESULT sample. This value is stored by the companion module and is used to address the destination RFIFO register when a RESULT_DATA is generated due to that ADC_CONV_RESULT sample. Note: In case the companion module only receives data from eQADC and does not return any result to eQADC RFIFOs, the MESSAGE_TAG field can be used as a destination router in the companion module. The eQADC module can send conversion results to sub-blocks in the companion module that are addressed by the MESSAGE_TAG value.
TAG[0:3]	Companion module tag bits This bit field is used to address the appropriate destination RFIFO in the EQADC for the accompanying RESULT_DATA bits. However, it can be used too in the companion module to select the destination of the conversion result data. In eQADC application, this is used to address the appropriate RFIFO in the eQADC block. In this case, the possible values are only from 0000 to 0101.
ADC_CONV_RESULT[0:15]	ADC Conversion Result Data This bit field is the ADC conversion result data after passing through the calibration and formatting block.
RESULT_DATA[0:15]	Companion Module Result Data This bit field corresponds to the companion module data processing result to EQADC.

30.6.9.2 PSI Transmitter / Write Section

The transmission sub-block formats the data bus from RFIFO control sub-block to send to the PSI slave wdata bus. The transmission data is registered and its content is described in [Input / output signals description](#). The transmission has higher priority than reception. This is done to avoid the use of memory to store transmission data and it is not used waiting time for transmission.

The destination companion module for the transmission data is obtained by decoding the DEST[0:3] and DEST2 bits. The not null decimal value of DEST[0:3] or DEST2 is used to set one or two corresponding module enable signals as already described. For example, DEST[0:3] value equal to 0x1 that corresponds to the decimal value 1 is going to set only the module enable 1. All other module enable from 2 to 15 are not set due to DEST.

However, using the DEST2 value there are cases that two module enables are set, as described in [Table 30-121](#) in [Alternate Configuration 1-14 Control Registers \(ADC_ACR1-14\)](#).

30.6.9.3 PSI Receiver / Read Section

The receiver sub-block receives data from some companion module using the PSI slave bus interface. The companion module sends a read request to EQADC using the DMA read request line. The PSI logic sends a read command if there is no transmission request. The received data has the structure described in [Input / output signals description](#).

The receiver section can receive read request from several companion modules at the same time. A prioritization scheme is used to decide which request should be served first. The request with lower index has higher priority.

Note

In case of receiving data (D1, D2) and timestamp (T1, T2) from two companion modules with the same TAG at almost the same time, the RFIFOx can receive the pair data/timestamp out of order due to the present prioritization logic in the PSI receiver. An expected sequence (D1 T1 D2 T2) can be changed to (D1 D2 T2 T1) if D2/T2 has higher priority than T2. [Table 30-125](#) presents some cases that presents this inversion.

30.6.10 Analog sub-block

30.6.10.1 Analog to digital converter (ADC)

30.6.10.1.1 ADC architecture

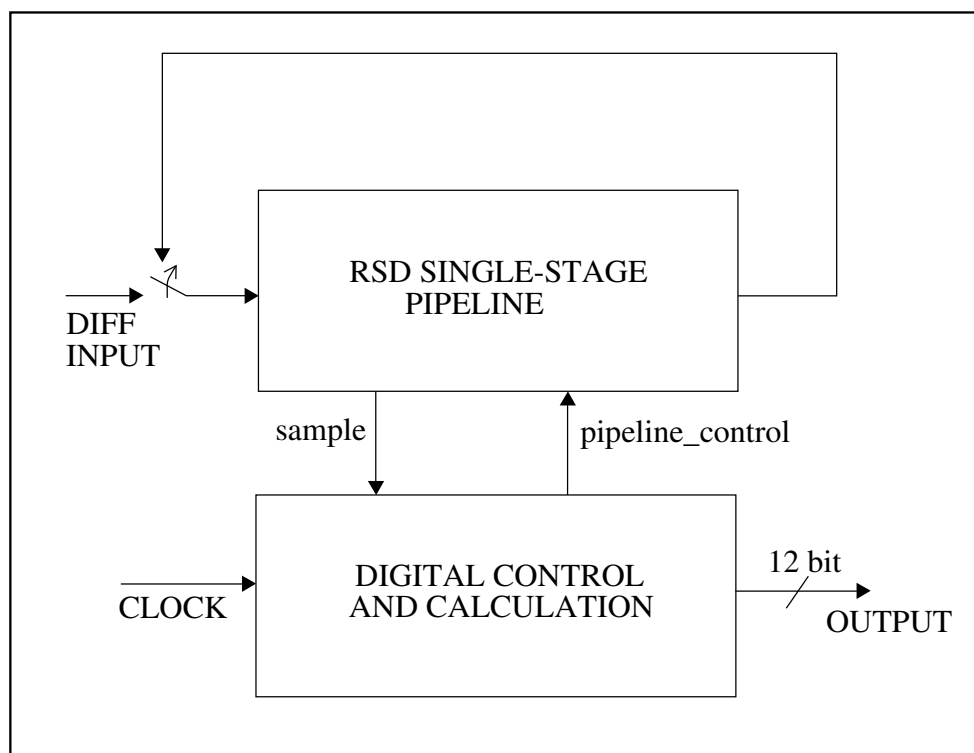


Figure 30-149. RSD ADC block diagram

The RSD Cyclic ADC consists of two main portions, the analog RSD Stage, and the digital control and calculation block, as shown in [Figure 30-149](#). To begin an analog to digital conversion, a differential input is passed into the analog RSD stage. The signal is passed through the RSD stage, and then from the RSD stage output, back to its input to be passed again. To complete a 12-bit conversion, the signal must pass through the RSD stage 12 times. For 10-bit and 8-bit resolution, the signal must pass 10 or 8 times through the RSD. Each time an input signal is read into the RSD stage, a digital sample is taken by the digital control/calculation block. The digital control/calculation block uses this sample to tell the analog block how to condition the signal. The digital block also saves each successive sample and adds them according to the RSD algorithm at the end of the entire conversion cycle.

30.6.10.1.2 RSD Overview

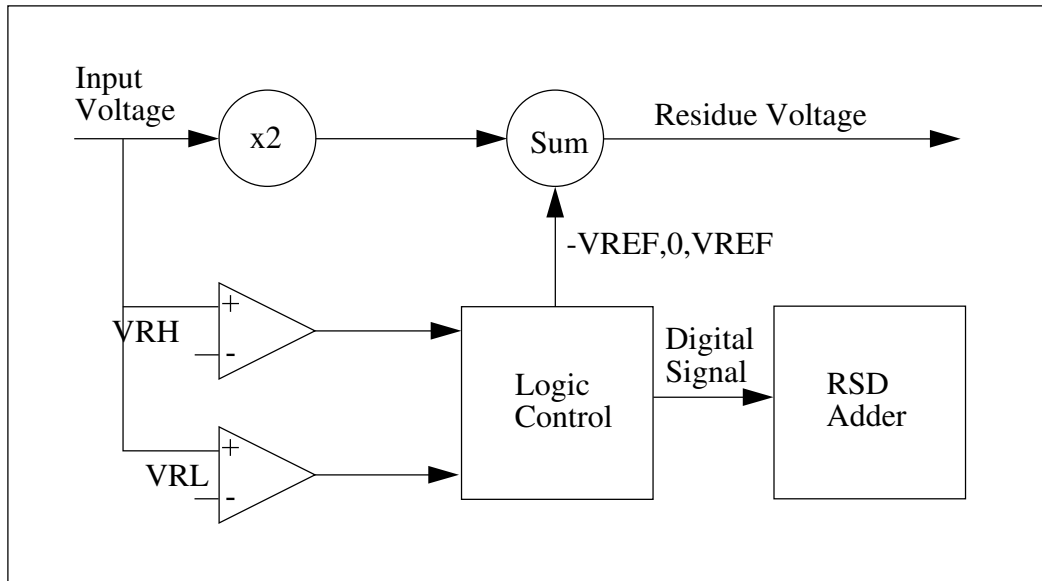


Figure 30-150. RSD stage block diagram

On each pass through the RSD stage, the input signal will be multiplied by exactly two, and summed with either $-V_{REF}$, 0, or V_{REF} , depending on the Logic Control. The Logic Control will determine $-V_{REF}$, 0, or V_{REF} depending on the two comparator inputs. As the Logic Control sets the summing operation, it also sends a digital value to the RSD adder. Each time an analog signal passes through the RSD single-stage, a digital value is collected by the RSD adder. At the end of an entire AD conversion cycle, the RSD adder uses these collected values to calculate the 12-bit/10-bit/8-bit digital output.

[Figure 30-151](#) shows the transfer function for the RSD stage. Note how the digital value (a, b) is dependent on the two comparator inputs.

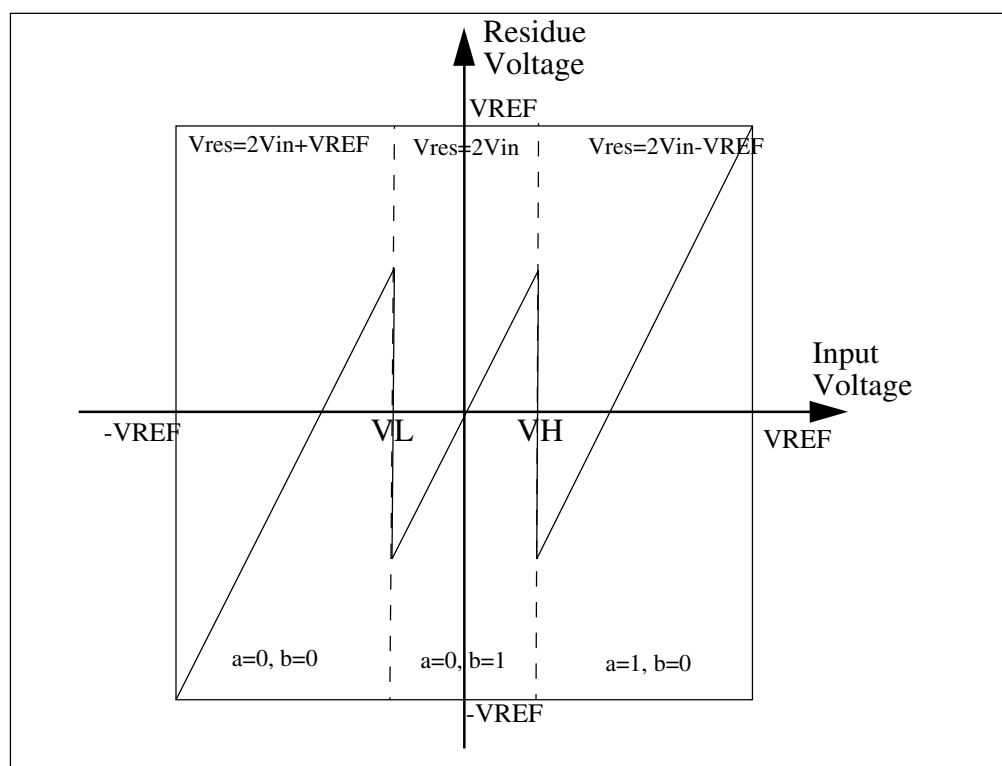


Figure 30-151. RSD Stage Transfer Function

In each pass through the RSD stage, the residue will be sent back to be the new input, and the digital signals, a and b , will be stored. For the 12-bit ADC, input signal is sampled during the input phase, and after each of the 12 passes through the RSD stage. Thus, 13 total a and b values are collected. Upon collecting all these values, they will be added according to the RSD algorithm to create the 12-bit digital representation of the original analog input. The bits are added in the following manner:

30.6.10.1.3 RSD adder

The array, $s1$ to $s12$, will be the digital output of the RSD ADC with $s1$ being the MSB and $s12$ being the LSB.

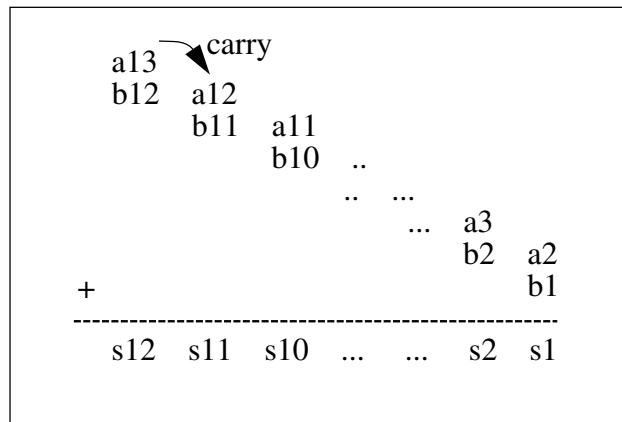


Figure 30-152. RSD adder

30.6.10.1.4 Variable Gain Amplification (VGA) for pre-gain

The VGA starts after sampling completes. It is enabled by a 2-bit signal PRE_GAIN described in [Alternate Configuration 1-14 Control Registers \(ADC_ACR1-14\)](#).

The ADC takes 2, 8, 64 or 128 clock cycles to do sampling which is selected by the LST[0:1] field in the conversion command message. After the sampling, if 2x VGA is enabled, there is a 2x gain stage without comparison before the regular conversion cycles. When 4x VGA is enabled, there are the 2x gain stage without comparison by 2 times before the normal conversion processing.

30.7 Initialization/application information

30.7.1 Multiple queues control setup example

This section provides an example of how to configure multiple CQueues. [Table 30-145](#) describes how each CQueue can be used for a different application. Also documented in this section are general guidelines on how to initialize the on-chip ADCs and how to configure the CQueues and the EQADC.

Table 30-145. Application of each CQueue

CQueue number	CQueue type	Running speed	Number of contiguous conversions	Example
0	Very fast burst time-based CQueue	every 2 μ s for 200 μ s; pause for 300 μ s and then repeat	2	Injector current profiling
1	Fast hardware-triggered CQueue	every 900 μ s	3	Current sensing of PWM controlled actuators

Table continues on the next page...

Table 30-145. Application of each CQueue (continued)

CQueue number	CQueue type	Running speed	Number of contiguous conversions	Example
2	Fast repetitive time-based CQueue	every 2 ms	8	Throttle position
3	Software-triggered CQueue	every 3.9 ms	3	Command triggered by software strategy
4	Repetitive angle-based CQueue	every 625 μ s	7	Airflow read every 30 degrees at 8000 RPM
5	Slow repetitive time-based CQueue	every 100 ms	10	Temperature sensors

30.7.1.1 EQADC initialization

The following steps provide an example about how to configure the EQADC controls and how to initialize the on-chip ADCs. In this example, all conversion commands will be transferred through CFIFO0.

1. Load all required configuration commands in the RAM in such way that they form a queue; this data structure will be referred below as CQueue0. [Figure 30-153](#) shows an example of a CQueue able to configure the on-chip ADCs. Although, this example uses the DMAC to store commands in CFIFO0, configuration commands could have also been directly written to the CFIFO0 push register.
2. Select source driving EQADC hardware trigger ports (ETRIG). Before proceeding to next step, allow some time (minimum of two system clocks - filter period is set to minimum after reset) so that the logic level at the source is filtered and reaches the EQADC control logic.

Note

ETRIG ports could be driven by an external pin or by the output port of other blocks in the device, such as timers. In order to avoid unexpected triggering of CFIFOs in hardware trigger modes, the source driving the ETRIG port must be selected and set to a known logic level before putting the CFIFOs into the WAITING FOR TRIGGER state.

The trigger filter bypass control inputs must be set considering the characteristics of the trigger signal. A particular case to assert the bypass control is when a device's internal signal with one clock width pulse is used.

3. Configure [EQADC External Trigger Digital Filter Register \(EQADC_ETDFR\)](#).
4. Configure the DMAC to transfer data from CQueue0 to CFIFO0 in the EQADC.
5. Configure [EQADC Interrupt and DMA Control Register 0 \(EQADC_IDCR0\)](#).
 - Set CFFS0 to configure the EQADC to generate a DMA request to load commands from CQueue 0 to the CFIFO0.
 - Set CFFE0 to enable the EQADC to generate a DMA request to transfer commands from CQueue0 to CFIFO0; Command transfers from the RAM to the CFIFO0 will start immediately.
 - Set EOQIE0 to enable the EQADC to generate an interrupt after transferring all of the commands of CQueue0 through CFIFO0.
6. Configure [EQADC CFIFO Control Register 0 \(EQADC_CFCR0\)](#).
 - Write 0b0001 to the MODE0 field in EQADC_CFCR0 to program CFIFO0 for software single-scan mode.
 - Write "1" to SSE0 to assert SSS0 and trigger CFIFO0.
7. Since CFIFO0 is in single-scan software mode and it is also the highest priority CFIFO, the EQADC starts to transfer configuration commands to the on-chip ADCs.
8. When all of the configuration commands have been transferred, CF0 in [EQADC FIFO and Interrupt Status Register \(EQADC_FISR_n\)](#) will be set. The EQADC generates a End of Queue interrupt. The initialization procedure is complete.

		CQueue in system memory
Command Address	0x0	Configuration Command to CBuffer0 - Ex: Write ADC0_CR
	0x1	Configuration Command to CBuffer0 - Ex: Write ADC_TSCR
	0x2	Configuration Command to CBuffer1 - Ex: Write ADC1_CR

Figure 30-153. Example of a CQueue configuring the on-chip ADCs

The initialization procedure described above does not generate ADC clocks that are in phase because the timing at which the ADC0/1_EN bits, in [ADC0/1 Control Registers \(ADC0_CR and ADC1_CR\)](#), are set is different. Below follows an example on how to simultaneously set these bits so that in-phase ADC clocks are generated. In this example, ADC0/1_CLK are configured to the same frequency.

1. Push an ADC0_CR write configuration command in CFIFO0 that enables ADC0 (ADC0_EN=1) and that sets the ADC0_CLK_PS to an appropriate value. For example, 0x80800801.
2. Push an ADC1_CR write configuration command in CFIFO1 that enables ADC1 (ADC1_EN=1) and that sets the ADC1_CLK_PS to an appropriate value. For example, 0x82800801.
3. Configure CFIFO0 and CFIFO1 to single scan software trigger mode and simultaneously trigger them by writing 0x04100410 to the EQADC_CFCR0 register - see [EQADC CFIFO Control Register 0 \(EQADC_CFCR0\)](#).

30.7.1.2 Configuring EQADC for applications

This section provides an example based on the applications in [Table 30-145](#). The example describes how to configure multiple CQueues to be used for those applications and provides a step-by-step procedure to configure the EQADC and the associated CQueue structures. In the example, the "Fast hardware-triggered CQueue", described on the second row of [Table 30-145](#), will have its commands transferred to CBuffer1; the conversion commands will be executed by ADC1. The generated results will be returned to RFIFO3 before being transferred to the RQueues in the RAM by the DMAC.

Note

There is no fixed relationship between CFIFOs and RFIFOs with the same number. The results of commands being transferred through CFIFO1 can be returned to any RFIFO, regardless of its number. The destination of a result is determined by the MESSAGE_TAG field of the command that requested the result. See [Message format in EQADC](#) for details.

Step One: Setup the CQueues and RQueues.

1. Load the RAM with configuration and conversion commands. [Figure 30-154](#) is an example of how CQueue1 commands should be set.

- Each trigger event will cause four commands to be executed. When the EQADC detects the Pause bit asserted, it will wait for another trigger to restart transferring commands from the CFIFO.
- At the end of the CQueue, the "EOQ" bit is asserted as shown in [Figure 30-154](#).
- Results will be returned to RFIFO3 as specified in the MESSAGE_TAG field of commands.

2. Reserve memory space for storing results.

		0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
Bit Name												ADC COMMAND																					
	EOQ	PAUSE	REP	RESERVED	ER	BN	RESERVED	MESSAGE TAG																									
CMD 1	0	0	0	0	0	1	0	0b0011	Conversion Command																								
CMD 2	0	0	0	0	0	1	0	0b0011	Conversion Command																								
CMD 3	0	0	0	0	0	1	0	0b0011	Conversion Command																								
CMD 4	0	1	0	0	0	1	0	0b0011 ¹	Configure peripheral device for next conversion sequence																								
CMD 5	0	0	0	0	0	1	0	0b0011	Conversion Command																								
CMD 6	0	0	0	0	0	1	0	0b0011	Conversion Command																								
CMD 7	0	0	0	0	0	1	0	0b0011	Conversion Command																								
CMD 8	0	1	0	0	0	1	0	0b0011 ¹	Configure peripheral device for next conversion sequence																								
				0				etc.																									
CMD EOQ	1	0	0	0	0	1	0	0b0011	EOQ Message																								
CFIFO Header											ADC Command																						

CFIFO Header

ADC Command

Fields LST, TSR, FMT, and CHANNEL_NUMBER are not showed for clarity {Statement}.

¹ MESSAGE_TAG field is only defined for read configuration commands {Statement}.

Figure 30-154. Example of CQueue commands

Step Two: Configure the DMAC to handle data transfers between the CQueues/RQueues in RAM and the CFIFOs/RFIFOs in the EQADC.

1. For transferring, set the source address of the DMAC to point to the start address of CQueue1. Set the destination address of the DMAC to point to EQADC_CFPR1. See [EQADC CFIFO Push Register \(EQADC_CFPRn\)](#).
2. For receiving, set the source address of the DMAC to point to EQADC_RFPR3. See [EQADC Result FIFO Pop Register \(EQADC_RFPRn\)](#). Set the destination address of the DMAC to point to the starting address of RQueue1.

Step Three: Configure the EQADC Control Registers.

3. Configure [EQADC Interrupt and DMA Control Register 0 \(EQADC_IDCR0\)](#).
 - Set EOQIE1 to enable the End of Queue Interrupt request.
 - Set CFFS1 and RFDS3 to configure the EQADC to generate DMA requests to push commands into CFIFO1 and to pop result data from RFIFO3.
 - Set CFINV1 to invalidate the contents of CFIFO1.
 - Set RFDE3 and CFFE1 to enable the EQADC to generate DMA requests. Command transfers from the RAM to the CFIFO1 will start immediately.
 - Set RFOIE3 to indicate if RFIFO3 overflows.
 - Set CFUIE1 to indicate if CFIFO1 underflows.
4. Configure MODE1 to continuous-scan rising edge external trigger mode in [EQADC CFIFO Control Register 0 \(EQADC_CFCR0\)](#).

Step Four: Command transfer to ADCs and Result data reception.

When an external rising edge event occurs for CFIFO1, the EQADC automatically will begin transferring commands from CFIFO1 when it becomes the highest priority CFIFO trying to send commands to CBuffer1. The received results will be placed in RFIFO3 and then moved to RQueue1 by the DMAC.

30.7.2 EQADC/DMAC interface

This section provides an overview about the EQADC/DMAC interface and general guidelines about how the DMAC should be configured in order for it to correctly transfer data between the queues in system memory and the EQADC FIFOs.

Note

Advanced DMACs provide more functionality than the ones discussed in this section.

30.7.2.1 CQueue/CFIFO transfers

In transfers involving CQueues and CFIFOs, the DMAC moves data from a queued source to a single destination as showed in Figure 30-155. The location of the data to be moved is indicated by the source address, and the final destination for that data, by the destination address. The DMAC contains a data structure containing these addresses and other parameters used in the control of data transfers. For every DMA request issued by the EQADC, the DMAC has to be configured to transfer a single command (32-bit data) from the CQueue, pointed to by the source address, to the CFIFO push register, pointed to by the destination address. After the service of a DMA request is completed, the source address has to be updated to point to the next valid command. The destination address remains unchanged. When the last command of a queue is transferred one of the following actions is recommended. Refer to the DMAC block guide for details about how this functionality is supported.

- The corresponding DMA channel should be disabled. This might be desirable for CFIFOs in single scan mode.
- The source address should be updated to pointed to a valid command which can be the first command in the queue that has just been transferred (cyclic queue), or the first command of any other CQueue. This is desirable for CFIFOs in continuous scan mode, and at some cases, for CFIFOs in single scan mode.

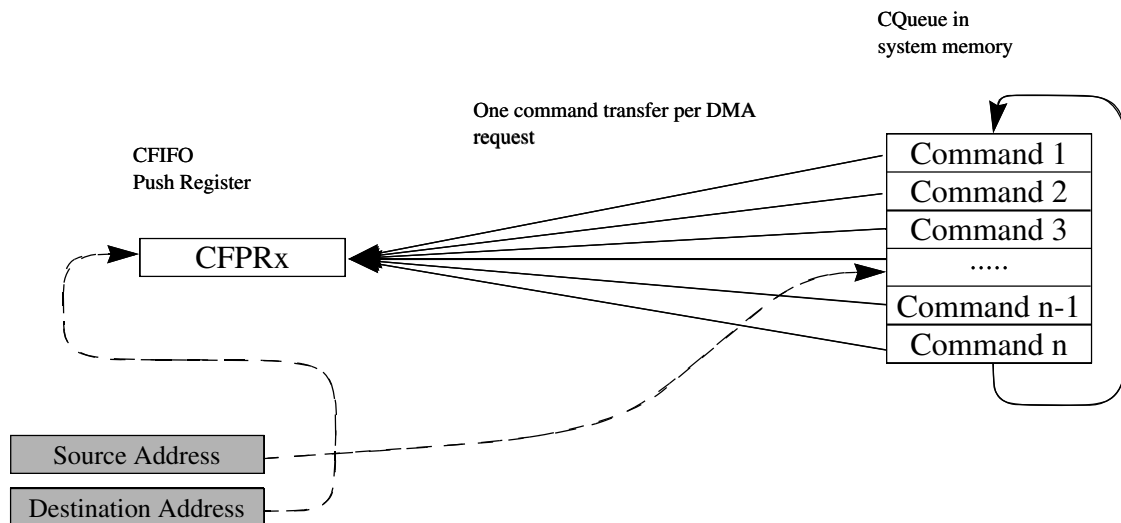


Figure 30-155. CQueue/CFIFO interface

30.7.2.2 RQueue/RFIFO transfers

In transfers involving RQueues and RFIFOs, the DMAC moves data from a single source to a queue destination as showed in [Figure 30-156](#). The location of the data to be moved is indicated by the source address, and the final destination for that data, by the destination address. For every DMA request issued by the EQADC, the DMAC has to be configured to transfer a single result (16-bit data), pointed to by the source address, from the RFIFO pop register to the RQueue, pointed to by the destination address. After the service of a DMA request is completed, the destination address has to be updated to point to the location where the next 16-bit result will be stored. The source address remains unchanged. When the last expected result is written to the RQueue, one of the following actions is recommended. Refer to the DMAC block guide for details about how this functionality is supported.

- The corresponding DMA channel should be disabled.
- The destination address should be updated pointed to the next location where new coming results are stored, which can be the first entry of the current RQueue (cyclic queue), or the beginning of a new RQueue.

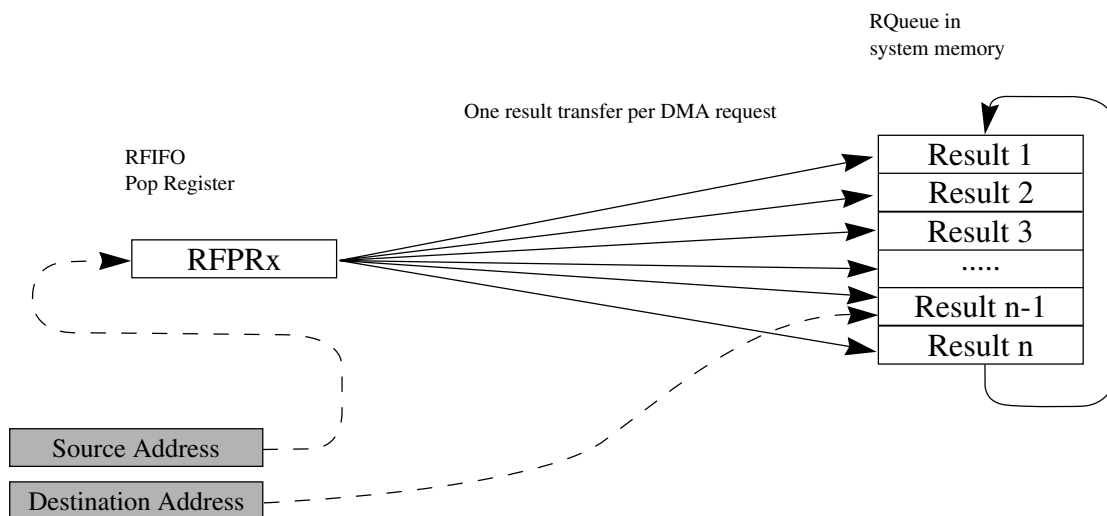


Figure 30-156. RQueue/RFIFO Interface

30.7.3 Sending immediate command setup example

In the EQADC, there is no immediate command register for sending a command immediately after writing to that register. However, a CFIFO can be configured to perform the same function as an immediate command register. The following steps illustrate how to configure CFIFO5 as an immediate command CFIFO. The results will be returned to RFIFO5.

1. Configure [EQADC Interrupt and DMA Control Register 0 \(EQADC_IDCR0\)](#).
 - Clear CFIFO Fill Enable5 (CFFE5 = 0) in EQADC_IDCR2.
 - Clear CFIFO Underflow Interrupt Enable5 (CFUIE5 = 0) in EQADC_IDCR2.
 - Clear RFDS5 to configure the EQADC to generate interrupt requests to pop result data from RFIFO5.
 - Set RFIFO Drain Enable5 (RFDE5 = 1) in EQADC_IDCR2.
2. Configure [EQADC CFIFO Control Register 0 \(EQADC_CFCR0\)](#).
 - Write "1" to CFINV5 in EQADC_FCR2. This will invalidate the contents of CFIFO5.
 - Set MODE5 to Continuous-Scan Software Trigger mode in EQADC_CFCR2.
3. To transfer a command, write it to EQADC CFIFO Push Register 5 (EQADC_CFPR5) with Message Tag = 0b0101. See [EQADC CFIFO Push Register \(EQADC_CFPRn\)](#).
4. Up to four commands can be queued in CFIFO5. Check the CFCTR5 status in EQADC_FISR5 before pushing another command to avoid overflowing the CFIFO. See [EQADC FIFO and Interrupt Status Register \(EQADC_FISRn\)](#).
5. When the EQADC receives a conversion result for RFIFO5, it generates an interrupt request. RFIFO Pop Register 5 (EQADC_RFPR5) can be popped to read the result. See [EQADC Result FIFO Pop Register \(EQADC_RFPRn\)](#).

30.7.4 Modifying queues

More CQueues may be needed than the six supported by the EQADC. These additional CQueues can be supported by interrupting command transfers from a configured CFIFO, even if it is TRIGGERED and transferring, modifying the corresponding CQueue in the RAM or associating another CQueue to it, and restarting the CFIFO. More details on disabling a CFIFO are described in [Disabled Mode](#).

1. Determine the resumption conditions when later resuming the scan of the CQueue at the point before it was modified.
 - Change MODEx in [EQADC CFIFO Control Register 0 \(EQADC_CFCR0\)](#) to Disabled. See [Disabled Mode](#) for a description of what happens when MODEx is changed to Disabled.

- Poll CFSx until it becomes IDLE in [EQADC CFIFO Status Register \(EQADC_CFSR\)](#).
 - Read and save TC_CFx in [EQADC CFIFO Transfer Counter Register 0 \(EQADC_CFTCR0\)](#) for later resuming the scan of the queue. The TC_CFx provides the point of resumption.
 - Since all result data may not have been stored in the appropriate RFIFO at the time MODEx is changed to disable, wait for all expected results to be stored in the RFIFO/RQueue before reconfiguring the DMAC to work with the modified RQueue. The number of results that must return can be estimated from the TC_CFx value obtained above.
2. Disable the DMAC from responding to the DMA request generated by CFFFx and RFDFx in [EQADC FIFO and Interrupt Status Register \(EQADC_FISRn\)](#).
 3. Write "0x0000" to the TC_CFx field.
 4. Load the new configuration and conversion commands into RAM. Configure the DMAC to support the new CQueue/RQueue, but do not configure it yet to respond to DMA requests from CFIFOx/RFIFOx.
 5. If necessary, modify [EQADC Interrupt and DMA Control Register 0 \(EQADC_IDCR0\)](#) to suit the modified CQueue.
 6. Write "1" to CFINVx in [EQADC CFIFO Control Register 0 \(EQADC_CFCR0\)](#) to invalidate the entries of CFIFOx. Perform any other modifications to EQADC_CFCR except changing MODEx from Disabled.
 7. Configure the DMAC to respond to DMA requests generated by CFFFx and RFDFx.
 8. Change MODEx to the modified CFIFO operation mode. Write "1" to SSEx to trigger CFIFOx if MODEx is software trigger.

30.7.5 CQueue and RQueues usage

[Figure 30-157](#) is an example of CQueue and RQueue usage. It shows the CQueue0 commands requesting results that will be stored in RQueue0 and RQueue1, and CQueue1 commands requesting results that will be stored only in RQueue1. Some Command Messages request data to be returned from the on-chip ADC, but some only configure them and do not request returning data. When a CQueue contains both write and read commands like CQueue0, the CQueue and RQueue entries will not be aligned; as shown in [Figure 30-157](#), the result for the second command of CQueue0 is the first entry of RQueue0. The figure also shows that CQueue and RQueue entries can also become

unaligned even if all commands in a CQueue request data as CQueue1. CQueue1 entries became unaligned to RQueue1 entries because a result requested by the forth CQueue0 command was sent to RQueue1. This happens because the system can be configured so that several CQueues can have its results sent to a single RQueue.

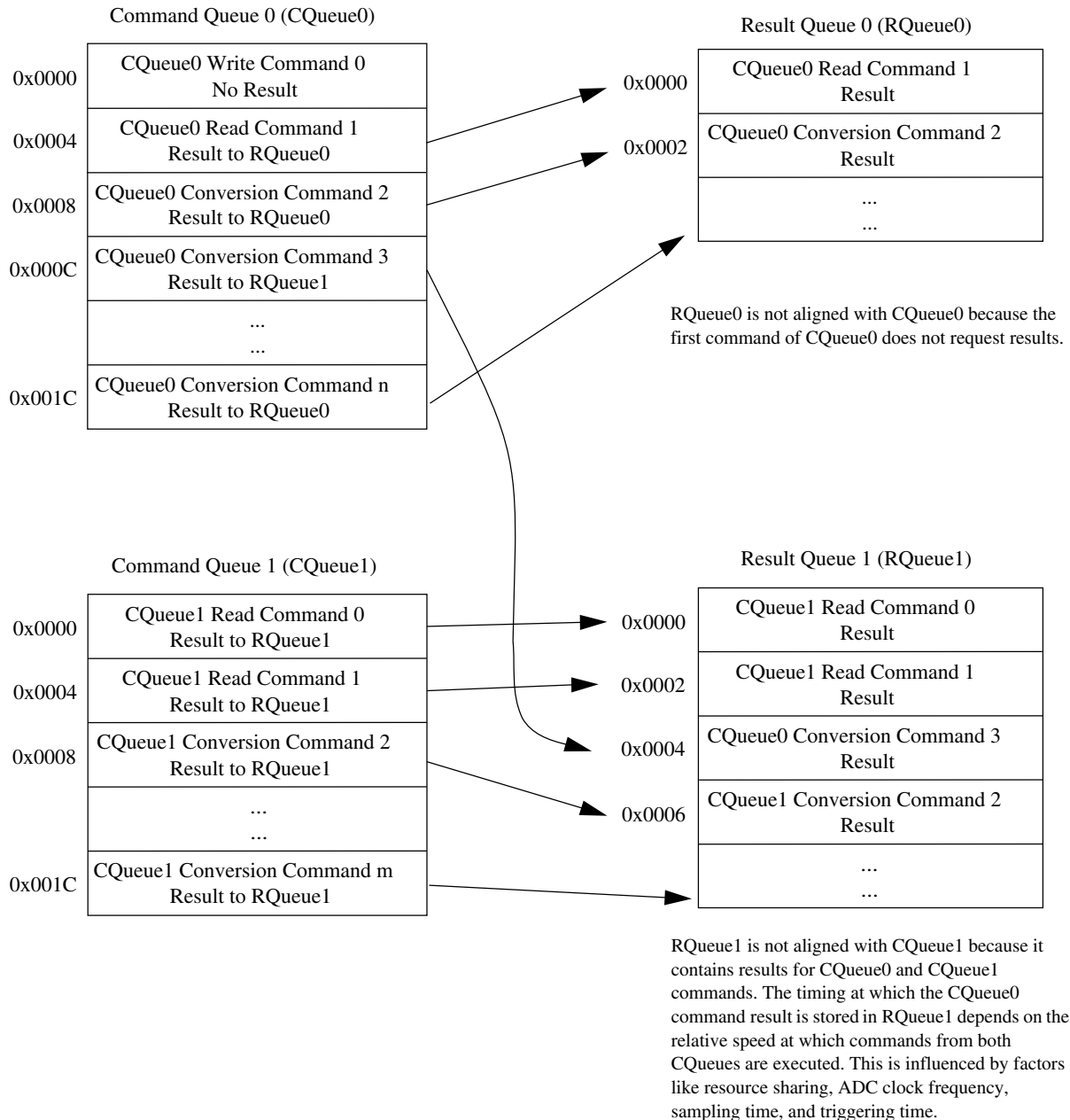


Figure 30-157. EQADC Command and Result Queues

30.7.6 ADC Result Calibration

The ADC result calibration process consists of two steps: determining the gain and offset calibration constants, and calibrating the raw results generated by the on-chip ADCs by solving the following equation discussed in [ADC calibration feature](#).

$$\text{CAL\_RES} = \text{GCC} * \text{RAW\_RES} + \text{OCC} + 2$$

Figure 30-158. Equation for CAL_RES

The calibration constants GCC and OCC can be calculated from equation [Figure 30-159](#), [Figure 30-160](#), and [Figure 30-161](#) provided that two pairs of expected (CAL_RES) and measured (RAW_RES) result values are available for two different input voltages. Most likely calibration points to be used are 25% VREF⁷ and 75% VREF since they are far apart but not too close to the end points of the full input voltage range. This allows for calculations of more representative calibration constants. The EQADC provides these voltages via channel numbers 43 and 44. The raw, uncalibrated results for these input voltages are obtained by converting these channels with conversion commands that have the CAL bit negated.

The transfer equations for when sampling these reference voltages are:

$$\text{CAL\_RES}_{75\% \text{VREF}} = \text{GCC} * \text{RAW\_RES}_{75\% \text{VREF}} + \text{OCC} + 2;$$

$$\text{CAL\_RES}_{25\% \text{VREF}} = \text{GCC} * \text{RAW\_RES}_{25\% \text{VREF}} + \text{OCC} + 2;$$

Thus;

$$\text{GCC} = (\text{CAL\_RES}_{75\% \text{VREF}} - \text{CAL\_RES}_{25\% \text{VREF}}) / (\text{RAW\_RES}_{75\% \text{VREF}} - \text{RAW\_RES}_{25\% \text{VREF}})$$

Figure 30-159. Equation for GCC

$$\text{OCC} = \text{CAL\_RES}_{75\% \text{VREF}} - \text{GCC} * \text{RAW\_RES}_{75\% \text{VREF}} - 2$$

Figure 30-160. Equation for OCC, form 1

or

$$\text{OCC} = \text{CAL\_RES}_{25\% \text{VREF}} - \text{GCC} * \text{RAW\_RES}_{25\% \text{VREF}} - 2$$

Figure 30-161. Equation for OCC, form 2

After being calculated, the GCC and OCC values must be written to ADC registers: [ADC0/1 Gain Calibration Constant Registers \(ADC0_GCCR and ADC1_GCCR\)](#) and [ADC0/1 Offset Calibration Constant Registers \(ADC0_OCCR and ADC1_OCCR\)](#) using write configuration commands.

7. VREF=VRH-VRL

The EQADC will automatically calibrate the results, according to equation [Figure 30-158](#), of every conversion command that has its CAL bit asserted using the GCC and OCC values stored in the ADC calibration registers.

30.7.6.1 MAC configuration procedure

The following steps illustrate how to configure the calibration hardware, namely, determining the values of the gain and offset calibration constants, and the writing of these constants to the calibration registers. The procedure below should be performed for ADC0 and for ADC1.

1. Convert channel 44 with a command that has its CAL bit negated and obtain the raw, uncalibrated result for 25%VREF (RAW_RES_{25%VREF}).
2. Convert channel 43 with a command that has its CAL bit negated and obtain the raw, uncalibrated result for 75%VREF (RAW_RES_{75%VREF}).
3. Since the expected values for the conversion of these voltages are known (CAL_RES_{25%VREF} and CAL_RES_{75%VREF}), GCC and OCC values can be calculated from [Figure 30-159](#) and [Figure 30-160](#) using these values, and the ones determined in steps 1 and 2.
4. Reformat GCC and OCC to the proper data formats as specified in [MAC unit and operand data format](#). GCC is an unsigned 15-bit fixed point value and OCC is a signed 14-bit value.
5. Write GCC value to [ADC0/1 Gain Calibration Constant Registers \(ADC0_GCCR and ADC1_GCCR\)](#), and OCC value to [ADC0/1 Offset Calibration Constant Registers \(ADC0_OCCR and ADC1_OCCR\)](#) using write configuration commands.

30.7.6.2 Example

The raw results obtained when sampling reference voltages 25%VREF and 75%VREF were, respectively, 3798 and 11592. The results that should have been obtained from the conversion of these reference voltages are, respectively, 4096 and 12288. Therefore, using equations [Figure 30-159](#) and [Figure 30-160](#), the gain and offset calibration constants are:

$$GCC = (12288 - 4096) / (11592 - 3798) = 1.05106492 \rightarrow 1.05102539 = 0x4344$$

$$OCC = 12288 - 1.05106492 * 11592 - 2 = 102.06 \rightarrow 102 = 0x0066$$

[Table 30-146](#) shows, for this particular case, examples of how the result values change according to GCC and OCC when result calibration is executed (CAL=1) and when it is not (CAL=0).

Table 30-146. Calibration example

Input voltage	Raw result (CAL=0)		Calibrated result (CAL=1)	
	Hexadecimal	Decimal	Hexadecimal	Decimal
25% VREF	0x0ED6	3798	0x1000	4095.794
75% VREF	0x2D48	11592	0x3000	12287.486

30.7.6.3 Quantization error reduction during calibration

[Figure 30-162](#) shows how the ADC transfer curve changes due to the addition of two to the MAC output during the calibration - see MAC output equation at [Overview](#) . The maximum absolute quantization error is reduced by half leading to an increase in accuracy.

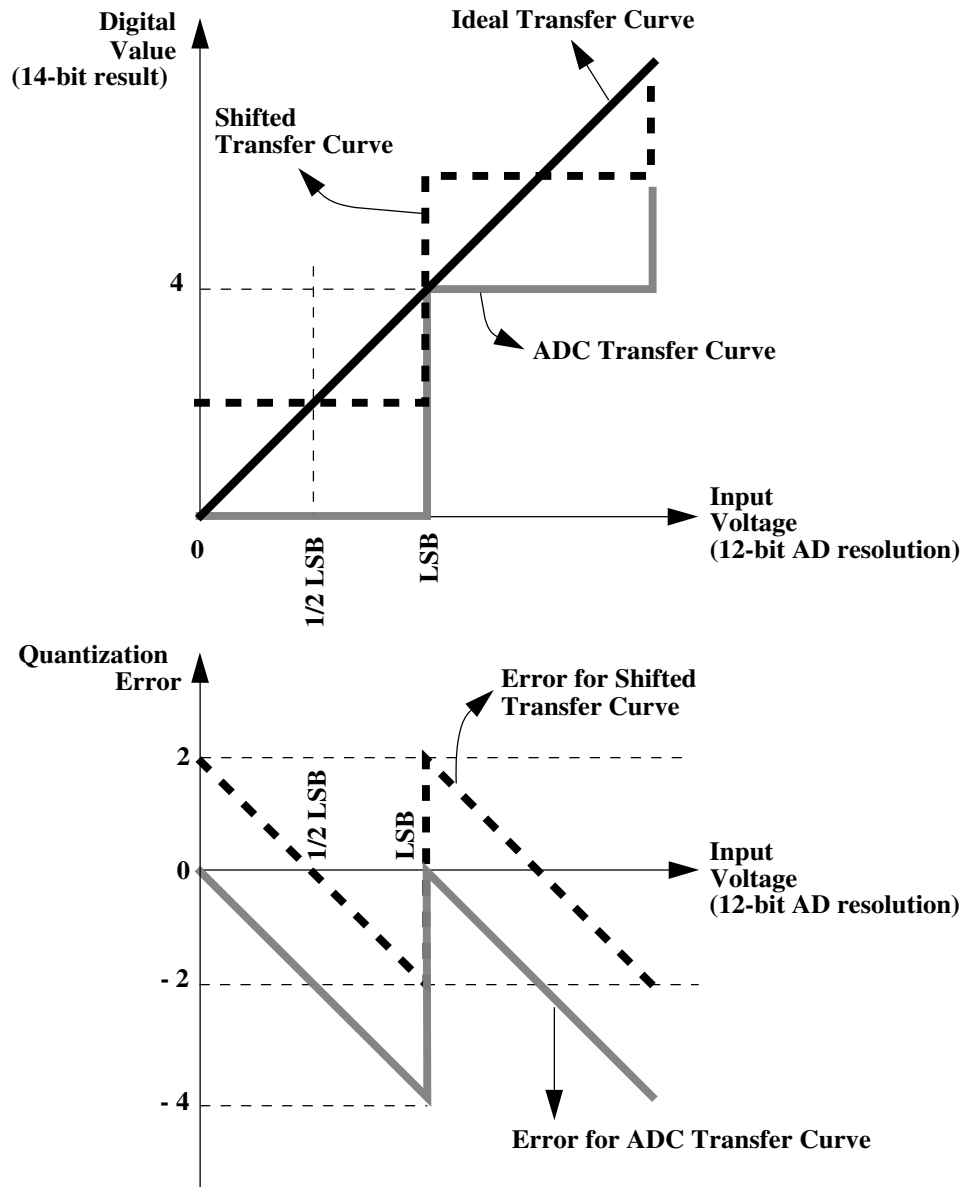


Figure 30-162. Quantization error reduction during calibration

30.7.7 EQADC versus QADC

This section describes how the EQADC upgrades the QADC functionality. The section also provides a comparison between the EQADC and QADC in terms of their functionality. This section targets the users familiar with terminology in QADC. [Figure 30-163](#) is an overview of a QADC. [Figure 30-164](#) is an overview of the EQADC system.

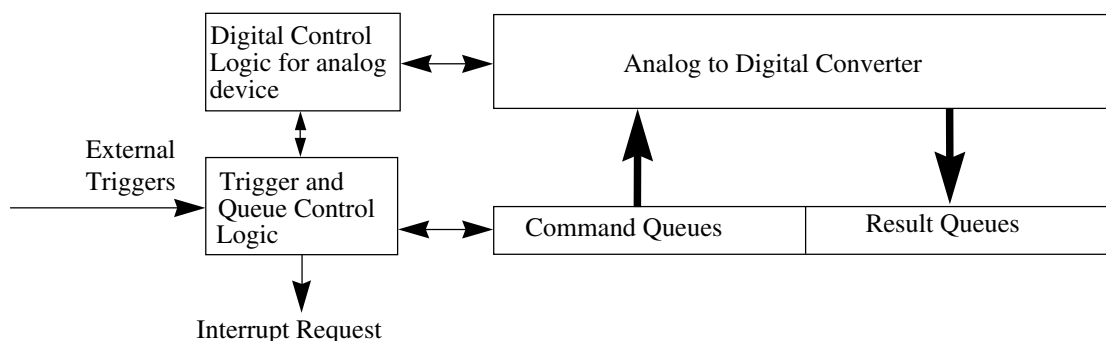


Figure 30-163. QADC overview

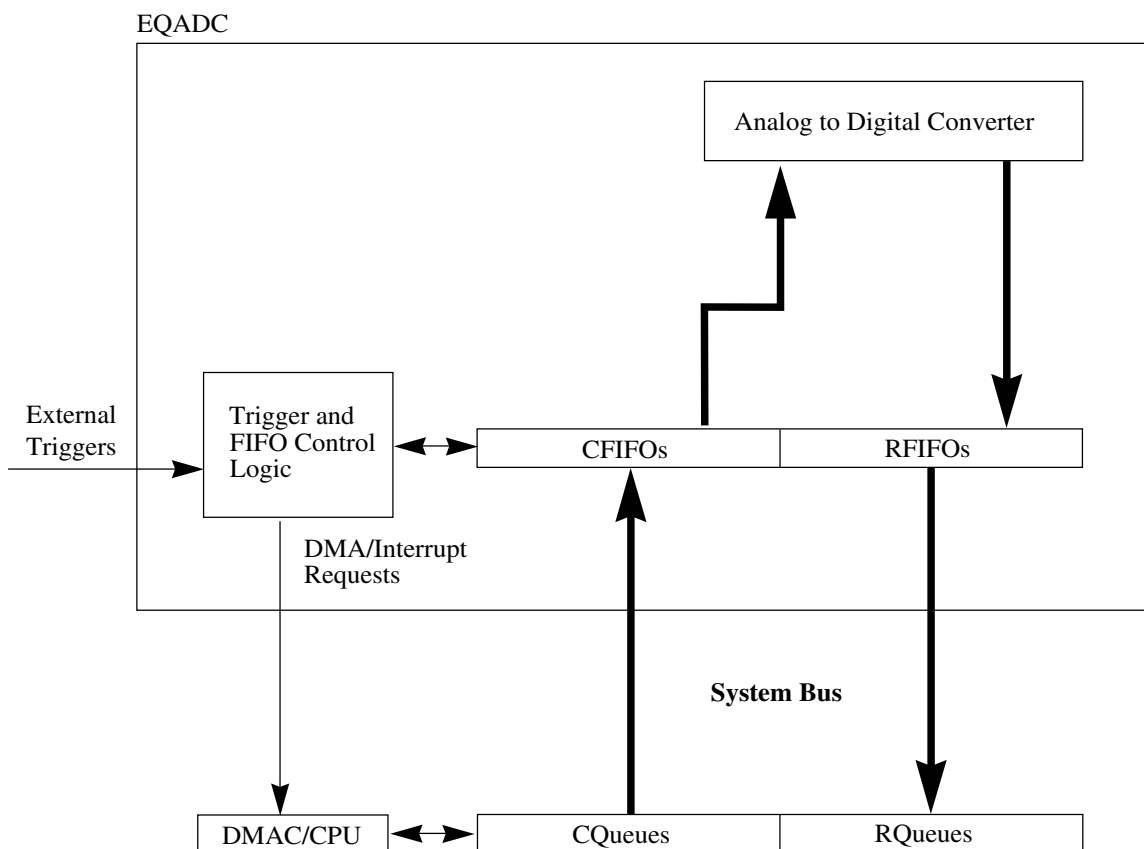


Figure 30-164. EQADC system overview

The EQADC system consists of queues in RAM, an external device, the EQADC, and on-chip ADCs. As compared with the QADC, the EQADC system requires extra hardware.

- A DMA or an MCU is required to move data between the EQADC's FIFOs and Queues in the system memory.

Since there are only FIFOs inside the EQADC, much of the terminology or use of the register names, register contents, and signals of the EQADC involve "FIFO" instead of "Queue". These register names, register contents, and signals are functionally equivalent to the "Queue" counterparts in the QADC. [Table 30-147](#) lists how the EQADC register, register contents, and signals are related to QADC.

Table 30-147. Terminology comparison between QADC and EQADC

QADC terminology	EQADC terminology	Function
CCW	Command Message	In the QADC, the hardware only executes conversion command words. In the EQADC, not all commands are conversion commands; some are configuration commands.
Queue Trigger	CFIFO Trigger	In the QADC, a trigger event is required to start the execution of a queue. In the EQADC, a trigger event is required to start command transfers from a CFIFO. When a CFIFO is TRIGGERED and transferring, commands are continuously moved from CQueues to CFIFOs. Thus, the trigger event initiates the "execution of a queue" indirectly.
Current Word Pointer Queue x (CWPQx)	Counter Value of Commands Transferred from Command FIFOx (TC_CFx)	In the QADC, CWPQx allows the last executed command on queue x to be determined. In the EQADC, the TC_CFx value allows the last transferred command on CQueue x to be determined.
Queue Pause Bit (P)	CFIFO Pause Bit	In the QADC, detecting a pause bit in the CCW will pause the queue execution. In the EQADC, detecting a pause bit in the Command will pause command transfers from a CFIFO.
Queue Operation Mode (MQx)	CFIFO Operation Mode (MODEx)	The EQADC supports all queue operation modes in the QADC except operation modes related to a periodic timer. A timer elsewhere in the system can provide the same functionality if it is connected to ETRIGx.
Queue Status (QS)	CFIFO Status (CFSx)	In the QADC, the Queue Status is read to check whether a queue is idle, active, paused, suspended, or trigger pending. In the EQADC, the CFIFO Status is read to check whether a queue is IDLE, WAITING FOR TRIGGER (idle or paused in QADC), or TRIGGERED (suspended or trigger pending in QADC). What CFIFO is currently "active" can be determined by reading the LCFTCBn field on the EQADC_CFSSR registers.

The EQADC and QADC also have similar procedures for the configuration or execution of applications. [Table 30-148](#) shows the steps required for the QADC versus the steps required for the EQADC system.

Table 30-148. Usage comparison between QADC and EQADC system

Procedure	QADC	EQADC System
Analog Control Configuration	Configure analog device by writing to the QADC registers.	Program configuration commands into command queues.
Prepare Scan Sequence	Program scan commands into command queues.	Program scan commands into command queues.
Queue Control Configuration	Write to the QADC Control Registers.	Write to the EQADC Control Registers.

Table continues on the next page...

Table 30-148. Usage comparison between QADC and EQADC system (continued)

Procedure	QADC	EQADC System
Data Transferred between Queues and Buffers	Not Required.	Program the DMAC or the CPU to handle the data transfer.
Queue Execution	Require Software or External Trigger events to start queue execution.	Require Software or External Trigger events to start command transfers from a CFIFO.

Chapter 31

Decimation Filter (DECFILTER)

31.1 Chip-specific DECFILTER information

This chip has 12 instances of the DECFILTER module.

31.1.1 DECFILTER instance interconnections

The instances of the DECFILTER module on this chip are interconnected.

31.1.1.1 DECFILTER connections for Cascade mode

The DECFILTER module instances on this chip can be configured to work in the Cascade mode of operation to obtain a more complex filtering function. The output result of a filter block is connected to the input of the next filter by means of a dedicated interface. The following table shows the instance connections.

Table 31-1. DECFILTER instance interconnections for Cascade mode

Cascade output from	Cascade input to
DECFILTER_A	DECFILTER_B
DECFILTER_B	DECFILTER_C
DECFILTER_C	DECFILTER_D
DECFILTER_D	DECFILTER_E
DECFILTER_E	DECFILTER_F
DECFILTER_F	DECFILTER_G
DECFILTER_G	DECFILTER_H
DECFILTER_H	DECFILTER_I
DECFILTER_I	DECFILTER_J
DECFILTER_J	DECFILTER_K
DECFILTER_K	DECFILTER_L
DECFILTER_L	DECFILTER_A

31.1.1.2 Stopping clock to DECFILTER instances

On this chip, the clock input can be stopped to all DECFILTER module instances collectively or to individual DECFILTER module instances. The following table summarizes the options.

Table 31-2. DECFILTER options to stop clock

Option	Program to 1	Note
Stop clock to all DECFILTER instances at once	SIU_HLT1[DECFIL]	The clock is stopped after all DECFILTER instances acknowledge the halt request.
Stop clock to individual DECFILTER instance	DECFILTER_MCR[MDIS]	—

31.1.2 DECFILTER connections to other modules

The DECFILTER module instances on this chip interact with the EQADC and eTPU modules.

31.1.2.1 DECFILTER connections with EQADCs

For information about DECFILTER connections with the EQADC modules, see:

- [Analog Modules Overview](#)
- [EQADC connections to other modules](#)
- [Routing conversion results to REACM2 and DECFILTER modules](#)

On this chip, the parallel side interfaces (PSIs) of the DECFILTER modules are connected to the EQADC modules:

- PSI0 inputs of all DECFILTER instances are connected to EQADC_A.
- PSI1 inputs of all DECFILTER instances are connected to EQADC_B.

31.1.2.2 DECFILTER connections with eTPUs

On this chip, select channels from eTPU module instances function as the Zero, Integrate, and Read inputs to the integrators of DECFILTER module instances. For a given DECFILTER integrator, one eTPU channel is routed to all three of the Zero, Integrate, and Read inputs.

In the SIU_DECFILn registers:

- Each ZSELx field selects the eTPU channel to route to the DECFILTER integrator's Zero, Integrate, and Read inputs.
- Each HSELx field selects the eTPU channel to route to the DECFILTER integrator's Halt input.

The following diagram shows the connections to DECFILTER_A as an example.

- The combined Zero, Integrate, and Read (ZIR) inputs to DECFILTER_A appear as the ZIRA signal.
- HALTA is the Halt input to DECFILTER_A.

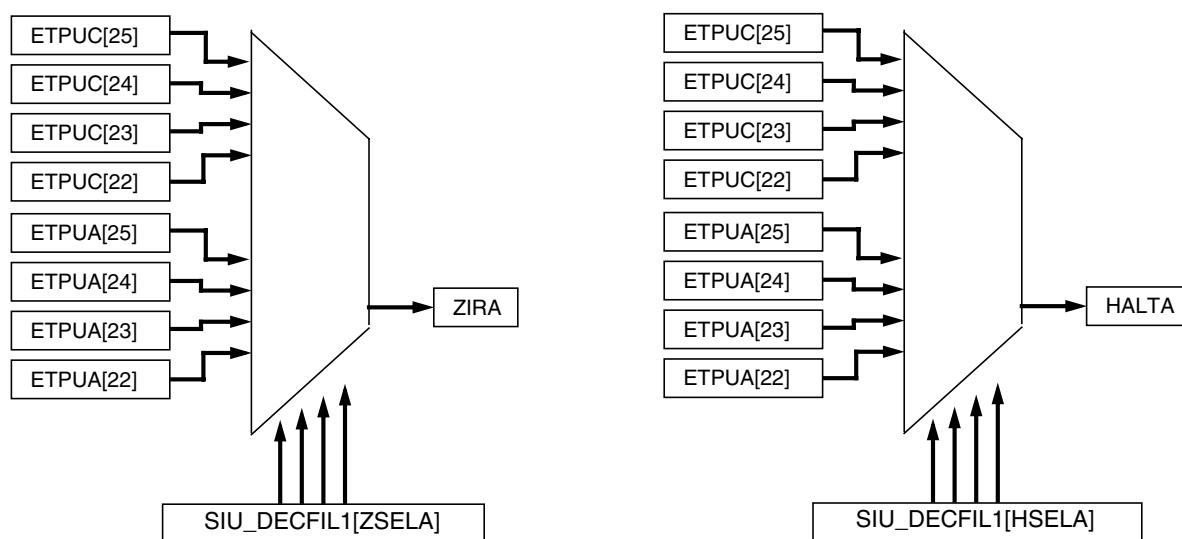


Figure 31-1. Example: Zero, Integrate, and Read (ZIR) and Halt connections between eTPU_A and DECFILTER_A

31.2 Introduction

This chapter describes the Decimation Filter block.

31.2.1 Overview

The decimation filter is a dedicated hardware block, designed to decimate fixed point sample conversion results, generated by a master block such as an Analog to Digital Converter (eQADC). Two dedicated parallel side interfaces (PSI0 and PSI1) provide bi-directional communication between two master blocks and the filter. A third interface is provided for use by the CPU, allowing setup of the filter parameters and read/write of the configuration registers and filter samples.

The Decimation Filter receives data samples from the master blocks (eQADCs) in the PSI RX sub-block. Each sample arrives at the decimation filter with an identifier tag and associated commands. The input information is decoded by the PSI RX and control logic sub-blocks. When receiving a filtering command, the data is transferred to the filter tap register's sub-block and is processed by the filter using the MAC (Multiply and Accumulate) unit, the coefficient register, and the control logic sub-blocks. The result is then returned to the master block by the PSI TX sub-block, accompanied by the corresponding tag information that provides an address for the data.

To summarize normal mode, the decimation filter works as a slave block on the third slave-bus line, and there are up to two PSI master blocks such as the ADC to send and read data. This is illustrated in the application example in [Application Information](#).

Note

The normal use is to receive the data stream from one specific PSI and transmit the results using one specific PSI. Input and output timing care is necessary if data is received from both PSI and output results are sent to both PSI alternately.

The decimation filter can also function in a standalone mode. In this mode, the input data is supplied and the output results are read by the chip core processor (CPU) using status and interrupt signals or DMA requests.

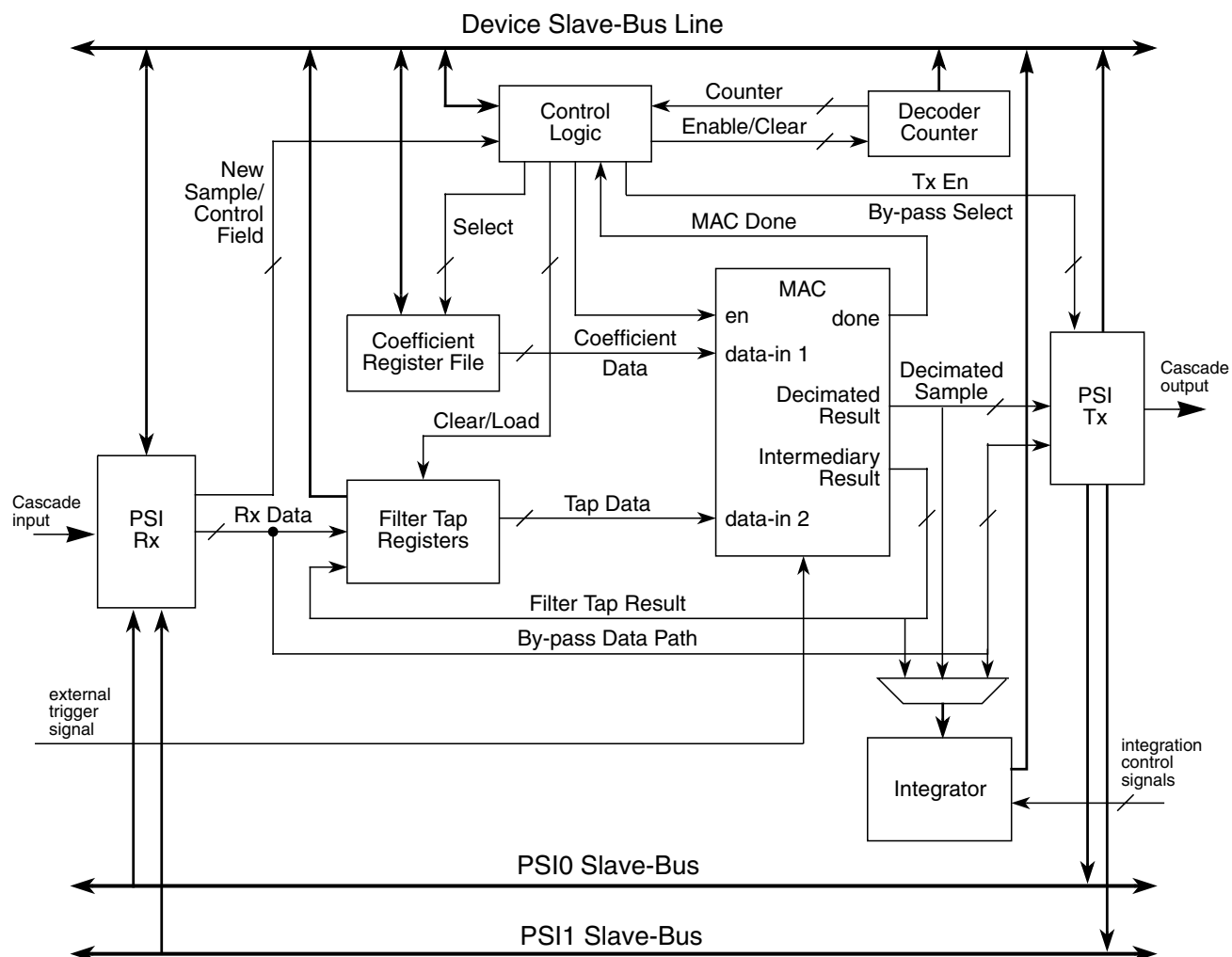
Mixed modes are also provided, allowing input data fed from the PSI interfaces, and output results read by the CPU or DMA, or input fed from CPU or DMA and output directed to a selected PSI interface.

An integrator unit independently accumulates the values of filter outputs. The integration can be restricted to time windows defined by hardware signals or software.

Two or more decimation filter blocks can also be configured to work in the cascade mode of operation to obtain a more complex filtering function. The output result of a filter block is connected to the input of the next filter by means of a dedicated interface.

All signals in the interface are generated in the system clock domain.

[Figure 31-2](#) shows the block diagram for the decimation filter.



31.2.2 Features

The Decimation Filter block includes these distinctive features:

- Selectable 4th order IIR filter, or an 8th order FIR filter
 - Input/output with 16-bit (fixed point) two's complement signed values
 - Internal taps with 16-bit (feed-forward portion of first IIR) and 24-bit (feedback portion) resolutions (fixed point) for two's complement signed value
 - 24-bit programmable filter coefficients (fixed point) for two's complement signed value
 - MAC unit with 51-bit fixed point accumulator
 - Convergent rounding methodology
 - Two's complement overflow or saturation selection
 - 58 clock cycles to process the input
- Implements two local slave-bus interface to two master blocks (e.g. the eQADC blocks)
- Input and output buffers with DMA capability
- Slave-bus interface to device
- Filter taps access for debug
- Filter initialization (flush) and stabilization (prefill) commands
- Timestamp support
- Decimation controlled by an internal counter or from an on-chip independent trigger signal (triggered output result).
- Integrator unit accumulates filter output values, signaled or absolute, with 32-bit resolution. The integrator can be controlled by software or hardware signals.
- Cascade of two or more individual blocks to compose a more complex filter.

31.2.3 Modes of operation

This section describes the operation modes of the Decimation Filter. The modes are selected using the following DECFILTER_MCR register fields: MDIS, FREN, FRZ, ISEL, MIXM, and CASCD (see [Module Configuration Register \(DECFILTER_MCR\)](#)).

The mode selection is summarized in the following table.

Table 31-3. Operation Mode Selection

Mode	MDIS	FREN, FRZ	ISEL	MIXM	CASCD
Normal	0	(0, 0)	0	0	00
Standalone		or	1	0	
PSI ¹ Input Mixed		(0, 1)	0	1	
PSI Output Mixed		or	1	1	
Cascade		(1, 0)	0 or 1	0	01 or 10 or 11
Freeze ²		1, 1	X	X	X
Low Power	1	X	X	X	X
Reserved	All other combinations				

1. The physical input / output can be PSI0 or PSI1.

2. Freeze mode can also be activated from outside the Decimation Filter, depending on the MCU, if FREN=1.

31.2.3.1 Normal mode

This is the default operational mode of the decimation filter block. It corresponds to the prefill/filter operation with input data supplied through any of the two PSI slave-bus interfaces (i.e. its input data is the ADC conversion result), with output going to the same or to the other PSI interface.

31.2.3.2 Standalone mode

Standalone mode differs from normal mode because the input data is not supplied by the master blocks through the PSI interfaces. Data is provided by the central processor using the device slave-bus interface or DMA interface signals. Once the data is filtered the decimated result is available in the Output Buffer register. The filter output is also consumed by a CPU or DMA mastering the same device slave-bus interface. This operation mode can be used to debug the filter stability or to decimate data in System RAM.

31.2.3.3 PSI Input Mixed mode

In this mode the input is selected from any of the two PSI slave-bus interfaces, but the output is directed to the device slave-bus interface, where it can be read by the CPU or DMA.

31.2.3.4 PSI Output Mixed mode

This mode works inverted from the PSI Input Mixed Mode: the input is selected from the deviceslave-bus interface, fed either by the CPU or DMA, and the output is directed to one of the two PSI slave-bus interfaces. If an eQADC is connected to one PSI interface, the output is directed to an RFIFO selected by the tag field in the DECFILTER_IB register (see [Interface Input Buffer \(DECFILTER_IB\)](#)).

31.2.3.5 Cascade mode

Cascade mode is a filter structure mode with two or more individual filter blocks connected in a chain to form a more complex filter function. The output result of the first block (head block) is connected to the input of the next block (middle or tail block) to be filtered again. More details in [Cascade mode description](#).

31.2.3.6 Low Power mode

Low power mode corresponds to the module disable mode or stop mode. In the module disable mode the two PSI slave-bus lines are disabled and it is not possible to enter Freeze mode. The system clock is stopped. In stop mode, the system clock is also stopped.

31.2.3.7 Freeze mode

Freeze mode is also known as debug mode. All filter action is frozen, either through software or by the hardware SoC debug request signal. If a freeze request comes when the filter is processing an input, it enters freeze mode only after the processing finishes.

31.3 External signal description

Note

The Decimation Filter does not provide metastability protection or filtering for these signals.

31.3.1 Decimation trigger signal

The decimation trigger signal is used to control the output of the decimation filter, allowing decimation to be driven externally. For more details, see [Triggered Output Result Description](#).

31.3.2 Integrator enable signal

The integrator enable signal is used to enable the hardware integrator. For more details, see [Integrator Enabling and Halting](#).

31.3.3 Integrator halt signal

The integrator halt signal is used to halt the hardware integrator. For more details, see [Integrator Enabling and Halting](#).

31.3.4 Integrator reset signal

The integrator reset signal is used to reset the hardware integrator. For more details, see [Integrator Reset](#).

31.3.5 Integrator output request signal

The integrator output request signal is used to update the integrator output result. For more details, see [Integrator Outputs](#).

31.4 Memory map and register definition

This section provides the memory maps and detailed descriptions of all registers. Accesses to reserved areas of the memory map can return a bus error.

This module communicates with three distinct slave-bus lines. One is related to the device integration and the two others are related to the PSI master blocks for data transfers. Below both memory maps are described.

The addresses of the Decimation Filter registers are specified as offsets from the module's base address, described in the following table. The registers allocated in this memory map are sufficient for a 4th order IIR filter implementation.

All registers are 32-bit wide.

NOTE

Reserved fields in the registers should not be written.

DECFILTER memory map

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
0	Module Configuration Register (DECFILTER_MCR)	32	R/W	See section	31.4.1/1284
4	Module Status Register (DECFILTER_MSR)	32	R/W	0000_0000h	31.4.2/1290
8	Module Extended Config Register (DECFILTER_MXCR)	32	R/W	0000_0000h	31.4.3/1294
C	Module Extended Status Register (DECFILTER_MXSR)	32	R/W	0000_0000h	31.4.4/1297
10	Interface Input Buffer (DECFILTER_IB)	32	R/W	0000_0000h	31.4.5/1300
14	Interface Output Buffer (DECFILTER_OB)	32	R	0000_0000h	31.4.6/1302
20	Filter Coefficient (DECFILTER_COEF0)	32	R/W	0000_0000h	31.4.7/1303
24	Filter Coefficient (DECFILTER_COEF1)	32	R/W	0000_0000h	31.4.7/1303
28	Filter Coefficient (DECFILTER_COEF2)	32	R/W	0000_0000h	31.4.7/1303
2C	Filter Coefficient (DECFILTER_COEF3)	32	R/W	0000_0000h	31.4.7/1303
30	Filter Coefficient (DECFILTER_COEF4)	32	R/W	0000_0000h	31.4.7/1303
34	Filter Coefficient (DECFILTER_COEF5)	32	R/W	0000_0000h	31.4.7/1303
38	Filter Coefficient (DECFILTER_COEF6)	32	R/W	0000_0000h	31.4.7/1303
3C	Filter Coefficient (DECFILTER_COEF7)	32	R/W	0000_0000h	31.4.7/1303
40	Filter Coefficient (DECFILTER_COEF8)	32	R/W	0000_0000h	31.4.7/1303
78	Filter TAP (DECFILTER_TAP0)	32	R	0000_0000h	31.4.8/1303
7C	Filter TAP (DECFILTER_TAP1)	32	R	0000_0000h	31.4.8/1303
80	Filter TAP (DECFILTER_TAP2)	32	R	0000_0000h	31.4.8/1303
84	Filter TAP (DECFILTER_TAP3)	32	R	0000_0000h	31.4.8/1303

Table continues on the next page...

DECFILTER memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
88	Filter TAP (DECFILTER_TAP4)	32	R	0000_0000h	31.4.8/1303
8C	Filter TAP (DECFILTER_TAP5)	32	R	0000_0000h	31.4.8/1303
90	Filter TAP (DECFILTER_TAP6)	32	R	0000_0000h	31.4.8/1303
94	Filter TAP (DECFILTER_TAP7)	32	R	0000_0000h	31.4.8/1303
D0	Enhanced Debug Input Data (DECFILTER_EDID)	32	R	0000_0000h	31.4.9/1304
E0	Final Integer Value Register (DECFILTER_FINTVAL)	32	R	0000_0000h	31.4.10/1305
E4	Final Integer Count Register (DECFILTER_FINTCNT)	32	R	0000_0000h	31.4.11/1305
E8	Current Integer Value Register (DECFILTER_CINTVAL)	32	R	0000_0000h	31.4.12/1306
EC	Current Integer Count Register (DECFILTER_CINTCNT)	32	R	0000_0000h	31.4.13/1306

31.4.1 Module Configuration Register (DECFILTER_MCR)

The Decimation Filter module configuration register provides configuration control bits for the Decimation Filter internal logic.

NOTE

One must not modify this register contents when the status bit BSY is set, except for fields FREN, FRZ, and IDIS. To guarantee that BSY does not set during the read-modify-write operation, set IDIS=1 and wait for BSY=0 beforehand.

The following table describes the device interface behavior determined by the combination of ISEL and MIXM bits of this register.

Table 31-6. ISEL/MIXM Definition — Read/Write from/to Input/Output Buffers

I S E L	MIX M	Mode	Operation	Devicelave-bus Interface		PSI slave-bus interfaces	
				Input Buffer	Output Buffer	Input Buffer	Output Buffer

Table continues on the next page...

Table 31-6. ISEL/MIXM Definition — Read/Write from/to Input/Output Buffers (continued)

I S E L	MIX M	Mode	Operation	Devicelave-bus Interface		PSI slave-bus interfaces	
0	0	Normal	Read	Always, by DMA or interrupt ¹ request if EDME=1	Always, no DMA or interrupt	forbidden, write only	By decfil request ²
	1	PSI Input Mixed			Always, issues DMA or interrupt ¹		Disabled
	0	Normal	Write	Forbidden	no effect, read only	Enabled	(read only)
	1	PSI Input Mixed					
1	0	Standalone	Read	Always, no DMA or interrupt	Always, issues DMA or interrupt ¹	forbidden, write only	Disabled
	1	PSI Output Mixed			Always, no DMA or interrupt		By decfil request ²
	0	Standalone	Write	Enabled, by DMA or interrupt request ¹	no effect, read only	Forbidden	read only
	1	PSI Output Mixed					

1. bit DSEL selects between interrupt or DMA request
2. Decimation filter issues a read request to the master block

NOTE

ISEL completely selects the output when the filter is configured as cascade tail, and is ignored when it is configured as cascade middle.

ISEL must not be modified during the filter operation (when the status bit BSY is set). In addition, the interface not selected by ISEL must not be used to write into the input buffer.

Address: 0h base + 0h offset = 0h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R			0		0						0			0		
W	MDIS	FREN		FRZ	SPES			CASCD	IDEN	ODEN	ERREN		FTYPE			SCAL
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Memory map and register definition

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R																
W																
Reset	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

DECFILTER_MCR field descriptions

Field	Description
0 MDIS	Module Disable The MDIS bit puts the Decimation Filter in low power mode. Communication through the two PSI slave-bus Interfaces are ignored in this mode. Writes to the configuration register are allowed with the exception of writes to the FREN and SRES bits, which are ignored. Writes to the Coefficient registers are also allowed. The Decimation Filter cannot enter Freeze mode once in disable mode. Once the module is disabled it no longer receives the system clock. 0 Normal Mode 1 Low Power Mode
1 FREN	Freeze Enable The FREN bit enables the Decimation Filter to enter freeze mode if the SoC debug request signal or the FRZ bit is asserted. See Freeze Mode Description for more details. 0 Decimation Filter Freeze mode disabled 1 Decimation Filter Freeze mode enabled
2 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
3 FRZ	Freeze Mode The FRZ bit controls the freeze mode of the Decimation Filter. For this bit to take effect the FREN freeze enable bit also needs to be asserted. While in freeze mode the MAC operations are halted. See Freeze Mode Description for more details. 0 Decimation Filter in Normal Mode 1 Decimation Filter in Freeze Mode
4 SRES	Software-reset bit The SRES is a self-negated bit which provides the CPU with the capability to initialize the Decimation Filter through the slave-bus interface. This bit always reads as zero. See Soft-Reset Command Description for more details. 0 No action 1 Software-Reset
5–6 CASCD	Cascade Mode Configuration The CASCD[1:0] bit field configures the block to work in cascade mode of operation. For more details about the cascade mode, see Cascade mode description.

Table continues on the next page...

DECFILTER_MCR field descriptions (continued)

Field	Description
	NOTE: Any change to this field must follow the procedure described in Cascade Freeze, Stop, and Configuration Change Procedures. 00 No cascade mode (single block) 01 Cascade Mode, Head block config. 10 Cascade Mode, Tail block config. 11 Cascade Mode, Middle block config.
7 IDEN	Input Data Interrupt Enable The IDEN bit enables the Decimation Filter to generate interrupt requests on all new input data written to the Interface Input Buffer register or Input/Output Buffers register. 0 Input Data Interrupt Disabled 1 Input Data Interrupt Enabled
8 ODEN	Output Data Interrupt Enable The ODEN bit enables the Decimation Filter to generate interrupt requests on all new data written to the filter Output buffer. It is independent of how ISEL and MIXM are set. 0 Output Data Interrupt Disabled 1 Output Data Interrupt Enabled
9 ERREN	Error Interrupt Enable The ERREN bit enables the Decimation Filter to generate interrupt requests based on the assertion of the DECFILTER_MSR register error flags OVF, DIVR, SVR, OVR, or IVR. 0 Error Interrupts Disabled 1 Error Interrupts Enabled
10 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
11–12 FTYPE	Filter Type Selection bits The FTYPE[1:0] bits select the filter type. NOTE: Bypass must not be configured in cascade mode (see field CASCD). 00 Filter Bypass In Bypass configuration the filter is disabled. 01 IIR Filter - 1 x 4th order 10 FIR Filter - 1 x 8th order 11 reserved
13 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
14–15 SCAL	Filter Scaling Factor The SCAL[1:0] bit field selects the scaling factor used by the filter algorithm. 00 Scaling Factor = 1 01 Scaling Factor = 4 10 Scaling Factor = 8 11 Scaling Factor = 16

Table continues on the next page...

DECFILTER_MCR field descriptions (continued)

Field	Description
16 IDIS	Input Disable The IDIS bit disables the block input, so that writes to the input buffer have no effect (either from the device slave-bus or from the two PSI interfaces), and input DMA or interrupt requests are not issued. Input disabling is needed to change the block configuration to or from cascade mode. See Saturation for more details. NOTE: IDIS resets in 1 (hardware reset only), and the module configuration procedures for previous versions (without IDIS bit) are upward compatible provided that the DECFILTER_MCR register is written with 0 into the IDIS position (previously reserved). 0 Input enabled 1 Input disabled
17 SAT	Saturation Enable The SAT bit enables the saturation of the filter output. See Saturation for more details. 0 Disables Saturation 1 Enable Saturation
18 ISEL	Input Selection The ISEL bit selects the source of input data to the Filter. Possible data sources are the master block (eQADC) of any PSI slave-bus interface, or the CPU/DMA on the device slave-bus interface. Each device slave-bus write to the Interface Input Buffer register or DMA transfer to the input buffer is interpreted as a new sample to be processed by the filter. The output interface used is the same as the one selected by ISEL if the output selection bit MIXM=0. When MIXM=1, the output selection (slave-bus or device slave-bus) is contrary to the input selection (see ISEL/MIXM Definition table and the MIXM bit definition), configuring a mixed mode operation. The slave-bus interface can always read the input/output buffers, however the PSI slave-bus interfaces can only read the output buffer by request of the decimation filter, in normal or input mixed modes. This behaviour is outlined in detail in Table 31-6 NOTE: There is no configuration register to select between the two PSI interfaces as the active one. Both are active when ISEL=0. 0 Filter input from any of the two PSI slave-bus interfaces 1 Filter input from the device slave-bus interface
19 MIXM	Mixed Mode The MIXM field selects the interface used for filter output, either device slave-bus or the PSI slave-bus, in relation to the interface selected by ISEL (see ISEL bit definition and ISEL/MIXM Definition table for more details). NOTE: MIXM must be set to 0 (zero) when the filter is configured as cascade mode. 0 Interface selected by ISEL is used for output, configuring normal or standalone mode 1 Interface NOT selected by ISEL is used for output, configuring mixed mode.
20–23 DEC_RATE	Decimation Rate Selection The DEC_RATE[3:0] field selects the decimation rate used by the Decimation Filter. The decimation rate defines the number of data samples from the master block that is required to generate one decimated result in the Decimation Filter output. 0000 No Decimation: one filter output for each sample input 0001 - 1111 One filter output for each (DEC_RATE+1) sample inputs

Table continues on the next page...

DECFILTER_MCR field descriptions (continued)

Field	Description
24 SDIE	Integrator Data Interrupt Enable The SDIE field enables output buffer interrupts due to integrator data result being ready (at registers DECFILTER_FINTVAL and DECFILTER_FINTCNT): 0 Integration ready does not cause an output interrupt. 1 Integration ready causes an output interrupt
25 DSEL	DMA Selection The DSEL bit determines whether the data transfers — to the input buffer (write to) and from the output buffer (read from) — are performed by DMA requests or by interrupt requests. This bit can also be active when PSI inputs are selected with Enhanced Debug (ISEL=0, EDME=1), in which case the input buffer generates read requests only (see Enhanced Debug Monitor Description). 0 Interrupt requests are generated 1 DMA requests are generated
26 IBIE	Input Buffer Interrupt Request Enable The IBIE bit enables the Decimation Filter to generate interrupt requests when: • device slave-bus input is selected (ISEL=1) and DSEL=0 when the input buffer is available to receive new data; • PSI inputs are selected with Enhanced debug (ISEL=0, EDME=1) and DSEL=0 when the input buffer has data to be read by the device CPU. 0 Input Buffer Interrupt Request Disabled 1 Input Buffer Interrupt Request Enabled
27 OBIE	Output Buffer Interrupt Request Enable The OBIE bit enables the Decimation Filter interrupt requests when outputs are directed to the device slave-bus (ISEL != MIXM) and DMA is not selected (DSEL=0). 0 Output Buffer Interrupt Request Disabled 1 Output Buffer Interrupt Request Enabled
28 EDME	Enhanced Debug Monitor Enable The EDME bit defines the enhanced debug monitor when input selection is from any PSI (ISEL=0). In this case, the raw data fed from the PSI Master blocks is also, in parallel, made available in the register DECFILTER_EDID (see Enhanced Debug Monitor Description), generating an input interrupt or DMA request. 0 Enhanced debug monitor enabled (read requests of input data from master block enabled) 1 Enhanced debug monitor disabled
29 TORE	Triggered Output Result Enable The TORE bit enables an input trigger signal to force the decimation filter to send the next result of the filter back to the master block. For more details, see Triggered Output Result Description. NOTE: TORE must only be asserted when PSI is selected as output (normal or PSI output mixed modes). TORE must not be asserted with the filter bypassed (FTYPE=00). 0 Output buffer update using an external signal is disabled 1 Output buffer update using an external signal is enabled

Table continues on the next page...

DECFILTER_MCR field descriptions (continued)

Field	Description
30–31 TMODE	Trigger Mode The TMODE field selects the way the trigger signal controls the output result sampling function enabled by the TORE bit. NOTE: The TMODE definition replaces, and is upward compatible with, the TRFE bit definition found in previous versions of the Decimation Filter. 00 output is posted at the rising edge of the trigger signal 01 output is posted whenever the trigger signal is a logical 0 10 output is posted at the falling edge of the trigger signal 11 output is posted whenever the trigger signal is a logical 1

1. bit DSEL selects between interrupt or DMA request
2. Decimation filter issues a read request to the master block

31.4.2 Module Status Register (DECFILTER_MSR)

Address: 0h base + 4h offset = 4h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	BSY	0	DEC_COUNTER				0	0	0	0	0	0	0	0	0	0
W							IDFC	ODFC		IBIC	OBIC		DIVRC	OVFC	OVRC	IVRC
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0						IDF	ODF	0	IBIF	OBIF	0	DIVR	OVF	OVR	IVR
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

DECFILTER_MSR field descriptions

Field	Description
0 BSY	Decimation Filter Busy indication

Table continues on the next page...

DECFILTER_MSR field descriptions (continued)

Field	Description
	The BSY bit indicates that the Decimation Filter is processing new input data from the master block in normal mode or from the device core in standalone mode. BSY is not asserted when the filter is disabled (FTYPE = 00). However, the BSY bit is asserted when the soft reset is executed. 0 Decimation Filter Idle 1 Decimation Filter Busy
1 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
2–5 DEC_COUNTER	Decimation Counter The DEC_COUNTER[3:0] field indicates the current value of the DEC_COUNTER Decimation Counter (see Figure 31-2), which counts the number of input data samples received by the Decimation Filter. When the value of this counter matches the DEC_RATE[3:0] Configuration Register field, one decimated result is generated and the DEC_COUNTER counter is re-initialized at zero. This register is cleared by a soft reset or a flush command.
6 IDFC	Input Data Flag Clear bit The IDFC bit clears the IDF Flag bit in the Status Register. This bit is self negated, therefore it is always read as zero. 0 No action 1 Clears IDF
7 ODFC	Output Data Flag Clear bit The ODFC bit clears the ODF Flag bit in the Status Register. This bit is self negated, therefore it is always read as zero. 0 No action 1 Clears ODF
8 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
9 IBIC	Input Buffer Interrupt Request Clear bit The IBIC bit clears the IBIF Flag bit in the Status Register. This bit is self negated, therefore it is always read as zero. 0 No action 1 Clears IBIF
10 OBIC	Output Buffer Interrupt Request Clear bit The OBIC bit clears the OBIF Flag bit in the Status Register. This bit is self negated, therefore it is always read as zero. 0 No action 1 Clears OBIF
11 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
12 DIVRC	DIVR Clear bit The DIVRC bit clears the DIVR Debug Filter Input Data Read Overrun indication bit in the Status Register. This bit is self negated, therefore it is always read as zero.

Table continues on the next page...

DECFILTER_MSR field descriptions (continued)

Field	Description
	0 No action 1 Clears DIVR
13 OVFC	OVF Clear bit The OVFC bit clears the OVF Output Overflow bit in the Status Register. This bit is self negated, therefore it is always read as zero. 0 No action 1 Clears OVF
14 OVRC	OVR Clear bit The OVRC bit clears the OVR Output Overrun bit in the Status Register. This bit is self negated, therefore it is always read as zero. 0 No action 1 Clears OVR
15 IVRC	IVR Clear bit The IVRC bit clears the IVR Filter Input Overrun indication bit in the Status Register. This bit is self negated, therefore it is always read as zero. 0 No action 1 Clears IVR
16–21 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
22 IDF	Input Data Flag The IDF bit flag indicates when new data is available at the DECFILTER_IB register or at the PSI bus data, input and output. This flag generates an Interrupt Request if enabled by the IDEN bit in the Configuration Register. This Flag is cleared by the IDFC Status bit or by a soft reset of the decimation filter. NOTE: OBS: This flag is not used for read / write requests. It is used only to announce the input data event. For read / write request flag, refer to IBIF. 0 Sample not received 1 New Sample received
23 ODF	Output Data Flag The ODF bit flag indicates when a new decimated sample is available at the DECFILTER_OB register or at the PSI bus data, input and output. This flag generates an Interrupt Request if enabled by the ODEN bit in the Configuration Register. This Flag is cleared by the ODFC Status bit or by a soft reset of the decimation filter. NOTE: OBS: This flag is not used for read requests. It is used only to announce the output data event. For read request flag, refer to OBIF. 0 No new Decimated Output Sample available 1 New Decimated Output Sample available
24 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

Table continues on the next page...

DECFILTER_MSR field descriptions (continued)

Field	Description
25 IBIF	Input Buffer Interrupt Request Flag The IBIF bit flag indicates that the input buffer DECFILTER_IB is available to be filled with new data when Enhanced Debug Monitor is off. In Enhanced Debug Monitor, it indicates the input buffer DECFILTER_IB was filled with a new sample and is ready to be read. IBIF assertion also asserts the interrupt signal when enabled by the IBIE bit in the Configuration Register when DMA is not selected (DSEL=0) and the input buffer requires access from the device slave-bus (ISEL != EDME). This Flag is cleared by the IBIC Status bit or by a soft reset of the decimation filter. 0 No action 1 New Sample is requested (ISEL = 1, EDME=0) or new sample is available in Enhanced Debug Monitor (ISEL=0, EDME=1).
26 OBIF	Output Buffer Interrupt Request Flag The OBIF bit flag indicates that a new decimated sample is available at the DECFILTER_OB register. This flag generates an Interrupt Request if enabled by the OBIE bit in the Configuration Register and with ISEL != MIXM and DSEL=0. This Flag is cleared by the OBIC Status bit or by a soft reset of the decimation filter. 0 No new Decimated Output available 1 New Decimated Output available
27 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
28 DIVR	Enhanced Debug Monitor Input Data Read Overrun The DIVR bit indicates that a received sample in the Filter Interface Input Register was overwritten by a new sample and was not read by the Core. This flag generates an Interrupt Request if enabled by the ERREN bit in the Configuration Register. This Flag is cleared by the DIVRC Status bit or by a soft reset of the decimation filter. 0 Input Data Read Overrun did not occur in Enhanced Debug monitor 1 Enhanced Debug Monitor Input Data Read Overrun occurred
29 OVF	Filter Overflow Flag The OVF bit indicates that an overflow occurred in the filtered sample result. This flag generates an Interrupt Request if enabled by the ERREN bit in the Configuration Register. This Flag is cleared by the OVFC Status bit or by a soft reset of the decimation filter. 0 No overflow 1 Overflow occurred
30 OVR	Output Interface Buffer Overrun The OVR bit indicates that a decimated sample was overwritten by a new sample in the Interface Output Buffer Register. This flag generates an Interrupt Request if enabled by the ERREN bit in the Configuration Register. This Flag is cleared by the OVRC Status bit or by a soft reset of the decimation filter. 0 No Output Overrun 1 Filter Output Overrun occurred
31 IVR	Input Interface Buffer Overrun The IVR bit indicates that a received sample in the Filter Interface Input Register was overwritten by a new sample. This was probably caused by a violation of the Decimation Filter maximum throughput. This flag generates an Interrupt Request if enabled by the ERREN bit in the Configuration Register. This Flag is cleared by the IVRC Status bit or by a soft reset of the decimation filter.

Table continues on the next page...

DECFILTER_MSR field descriptions (continued)

Field	Description
	NOTE: IVR does not set due to input register writes when input is disabled (DECFILTER_MCR bit IDIS=1). 0 Input Buffer Overrun did not occur 1 Input Buffer Overrun occurred

31.4.3 Module Extended Config Register (DECFILTER_MXCR)

Address: 0h base + 8h offset = 8h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R					0										0	0
W	SDMAE	SSIG	SSAT	SCSAT											SRQ	SZRO
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R		0			0				0				0			
W	SISEL		SZROSEL			SHLTSEL				SRQSEL				SENSEL		
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

DECFILTER_MXCR field descriptions

Field	Description
0 SDMAE	Integrator DMA Enable The SDMAE bit enables the DMA request when an integrator output is requested (see Integrator Outputs). NOTE: The DMA channel used is the same one used for filter outputs, and any configuration that generates DMA requests from both of those sources is not allowed. 0 integrator DMA request disabled 1 integrator DMA request enabled
1 SSIG	Integrator Signal operation selection

Table continues on the next page...

DECFILTER_MXCR field descriptions (continued)

Field	Description
	The SSIG bit defines how the filtered data signal is treated for integration: 0 integrator input takes the absolute value of filter output 1 integrator input takes signaled filter output
2 SSAT	Integrator Saturated operation selection The SSAT bit defines how the integrator accumulator behaves in case of an overflow. NOTE: In saturated operation the overflown integration sum holds the value 0xFFFFFFFF for absolute integration (SSIG=0), or values 0x7FFFFFFF (positive saturation) and 0x80000000 (negative saturation) for signaled integration (SSIG=1). NOTE: Non-saturated mode is not supported with signaled integration, therefore one must not configure SSIG=1 and SSAT=0. 0 integrator accumulator holds a modulo 2^{17} value (considering the 15-bit fractional part) on an overflow. 1 integrator accumulator saturates on an overflow
3 SCSAT	Integrator Counter Saturated operation selection The SCSAT bit defines how the integrator sample counter behaves in case of an overflow. 0 integrator sample counter holds a modulo 2^{32} value on an overflow. 1 integrator sample counter saturates on an overflow, holding a value of 0xFFFFFFFF.
4–13 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
14 SRQ	Integrator Output Request The SRQ bit is used to command the update of the integrator output, reflected in the registers DECIFILTER_FINTVAL and DECIFILTER_FINTCNT. It may also cause a DMAor interrupt request, depending on the DECIFILTER_MCR bit SDIE and DECIFILTER_MXCR bit SDMAE. This is a write-only bit, so reads always return 0. For more details see Integrator Reset. 0 no integrator output update request 1 requests integrator output update
15 SZRO	Integrator Zero The SZRO bit is used to zero the integrator sum. This is a write-only bit, reads always return 0. For more details see Integrator Reset. NOTE: If bits SRQ and SZRO are both written 1 at the same time, the integrator is reset only after the registers DECIFILTER_FINTVAL and DECIFILTER_FINTCNT are updated. 0 does not zero integrator sum 1 zeroes integrator sum
16 SISEL	Integrator Input Selection The SISEL bit selects the input of the integrator. For more details see Integrator Inputs. 0 decimated filter outputs feed the integrator 1 filter outputs before the decimation feed the integrator
17 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

Table continues on the next page...

DECFILTER_MXCR field descriptions (continued)

Field	Description
18–19 SZROSEL	Integrator Zero Control Mode Selection The SZROSEL field defines the use of the integrator zero hardware input signal. For more details see Integrator Reset. 00 hardware integrator zero request disabled 01 integrator zero on toggle of hardware signal 10 integrator zero on rising edge of hardware signal 11 integrator zero on falling edge of hardware signal
20–21 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
22–23 SHLTSEL	Integrator Halt Control Selection The SHLTSEL field defines the integrator halting mechanism. When the integrator is halted, the integration accumulator remains unaltered on filter outputs independently of the enabling selected by SENSEL. For more details see Integrator Enabling and Halting. 00 hardware halt control signal disabled 01 integrator halted, independently of the hardware signal 10 integrator halted when signal is at logical 0 11 integrator halted when signal is at logical 1
24 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
25–27 SRQSEL	Integrator Output Read Request Mode Selection The SRQSEL field defines the use of the integrator output request hardware input signal. An integrator output request updates the registers DECFILTER_FINTVAL and DECFILTER_FINTCNT, also causing a DMA or interrupt request. Note that DMA or interrupt requests due to integrator output updates depend on the DECFILTER_MXCR bit SDMAE and DECFILTER_MCR bit SDIE. When continuous output is on, an integrator output request is issued whenever a new filter output is accumulated. For more details see Integrator Outputs. 000 hardware output request disabled 001 integrator output request on toggle of hardware signal 010 integrator output request on rising edge of hardware signal 011 integrator output request on falling edge of hardware signal 100 reserved 101 continuous output request on, independently of hardware signal 110 continuous output request on when signal is at logical 0 111 continuous output request on when signal is at logical 1
28–29 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
30–31 SENSEL	Integrator Enable Control Selection The SENSEL field defines the integrator enabling mechanism. When the integrator is enabled, filter outputs selected by the SISEL bit are added to the integration accumulator. When the integrator is disabled, the integration accumulator remains unaltered on filter outputs. For more details see Integrator Enabling and Halting. 00 integrator disabled, independently of the hardware enable control signal 01 integrator enabled, independently of the hardware signal

Table continues on the next page...

DECFILTER_MXCR field descriptions (continued)

Field	Description
10	integrator enabled when signal is at logical 0
11	integrator enabled when signal is at logical 1

31.4.4 Module Extended Status Register (DECFILTER_MXSR)

Address: 0h base + Ch offset = Ch

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R				0				0		0	0	0	0	0	0	0
W								SDFC			SSEC	SCEC		SSOVFC	SCOVFC	SVRC
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R					0			SDF		0	SSE	SCE	0	SSOVF	SCOVF	SVR
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

DECFILTER_MXSR field descriptions

Field	Description
0–6 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
7 SDFC	Integrator Output Data Flag Clear bit The SDFC bit clears the SDF Flag bit in the Status Register. This bit is self negated, therefore it is always read as zero. 0 No action 1 Clears SDF

Table continues on the next page...

DECFILTER_MXSR field descriptions (continued)

Field	Description
8–9 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
10 SSEC	Integrator Sum Exception Clear bit The SSEC bit clears the SSE flag bit in the Status Register. This bit is self negated, therefore it is always read as zero. 0 No action 1 Clears SSE
11 SCEC	Integrator Count Exception Clear bit The SCEC bit clears the SCE flag bit in the Status Register. This bit is self negated, therefore it is always read as zero. 0 No action 1 Clears SCE
12 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
13 SSOVFC	Integrator Sum Overflow Clear bit The SSOVFC bit clears the SSOVF Flag bit in the Status Register. This bit is self negated, therefore it is always read as zero. 0 No action 1 Clears SSOVF
14 SCOVFC	Integrator Count Overflow Clear bit The SCOVFC bit clears the SCOVF Flag bit in the Status Register. This bit is self negated, therefore it is always read as zero. 0 No action 1 Clears SCOVF
15 SVRC	SVR Clear bit The SVRC bit clears the SVR Integrator Data Overrun indication bit in the Status Register. This bit is self negated, therefore it is always read as zero. 0 No action 1 Clears SVR
16–22 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
23 SDF	Integrator Data Flag The SDF bit flag indicates when a new integrator result is available at the DECFILTER_FINTVAL register. This flag generates an Interrupt Request if enabled by the SDIE bit in the Configuration Register. This flag is cleared by the SDFC Status bit or by a soft reset of the decimation filter. 0 no new integrator result available 1 new integrator result available
24–25 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

Table continues on the next page...

DECFILTER_MXSR field descriptions (continued)

Field	Description
26 SSE	Integrator Sum Exception flag The SSE bit indicates an exceptional condition of the integrator accumulator. This flag generates an Interrupt Request if enabled by the DECFILTER_MCR bit ERREN, and it is cleared by the SSEC bit or by a soft reset. Integrator exceptions are defined in section "Integrator Exceptions". 0 No exception in the integrator accumulator. 1 Integrator accumulator exception.
27 SCE	Integrator Count Exception flag The SCE bit indicates an exceptional condition of the integrator counter. This flag generates an Interrupt Request if enabled by the DECFILTER_MCR bit ERREN, and it is cleared by the SSEC bit or by a soft reset. Integrator exceptions are defined in section "Integrator Exceptions". 0 No exception in the integrator counter. 1 Integrator counter exception.
28 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
29 SSOVF	Integrator Sum Overflow Flag The SSOVF bit indicates an overflow of the integrator accumulator. This flag is cleared by the SSOVFC bit or by a soft reset. NOTE: The SSOVF bit samples the integrator accumulator overflow condition when and only when either registers DECFILTER_FINTVAL or DECFILTER_CINTCNT are updated. Therefore, only one of the register pairs (DECFILTER_FINTVAL/DECFILTER_FINTCNT and DECFILTER_CINTVAL/DECFILTER_CINTCNT) must be used by the application, in order to avoid races. 0 No overflow in the integrator accumulator. 1 Integrator accumulator overflow.
30 SCOVF	Integrator Count Overflow Flag The SCOVF bit flag indicates an overflow of the internal integrated sample counter. This flag is cleared by the SCOVFC bit or by a soft reset. NOTE: The SCOVF bit samples the integrator accumulator overflow condition when and only when either registers DECFILTER_FINTVAL or DECFILTER_CINTCNT are updated. Therefore, only one of the register pairs (DECFILTER_FINTVAL/DECFILTER_FINTCNT and DECFILTER_CINTVAL/DECFILTER_CINTCNT) must be used by the application, in order to avoid races. 0 No overflow in the integrator sample counter. 1 Integrator sample counter overflow.
31 SVR	Integrator Data Overrun The SVR bit indicates that an integration value and count in the registers DECFILTER_FINTVAL and DECFILTER_FINTCNT was overwritten by a new integrator output request and was not read by the CPU or DMA. This flag generates an Interrupt Request if enabled by the ERREN bit in the Configuration Register. This Flag is cleared by the SVRC bit or by a soft reset. 0 Integrator Data Overrun did not occur. 1 Integrator Data Overrun occurred.

31.4.5 Interface Input Buffer (DECFILTER_IB)

The Input Buffer Register provides access to the Input buffer of the decimation filter when the filter is in the standalone or PSI output mixed modes of operation. Writes to this register are interpreted as requests to the Decimation Filter to process new sample data. Writes to this register when ISEL=0 are not allowed.

Address: 0h base + 10h offset = 10h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0			PSIOSEL	INTAG				0						PREFILL	FLUSH
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	INPBUF															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

DECFILTER_IB field descriptions

Field	Description
0–2 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
3 PSIOSEL	PSI Output Selection The PSIOSEL bit selects the output PSI to send result filter in PSI output mixed mode. In standalone mode this bit has no effect. For more details, see Return PSI (RPSI) and PSI Output Selector (PSIOSEL) . NOTE: The value of PSIOSEL should be constant for some data stream when in output mixed mode. Therefore, the output data is sent to a specific PSI and avoids undesired output overrun events. 0 PSI0 is selected to send result data (in PSI output mixed mode) 1 PSI1 is selected to send result data (in PSI output mixed mode)
4–7 INTAG	Decimation filter input tag bits The INTAG[3:0] bit field is defined as a selector signal and it is used to identify different destinations for the INPBUF[15:0] data. When the PSI master block is an eQADC, it is used in PSI output mixed mode to address the appropriate RFIFO in the eQADC block.
8–13 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

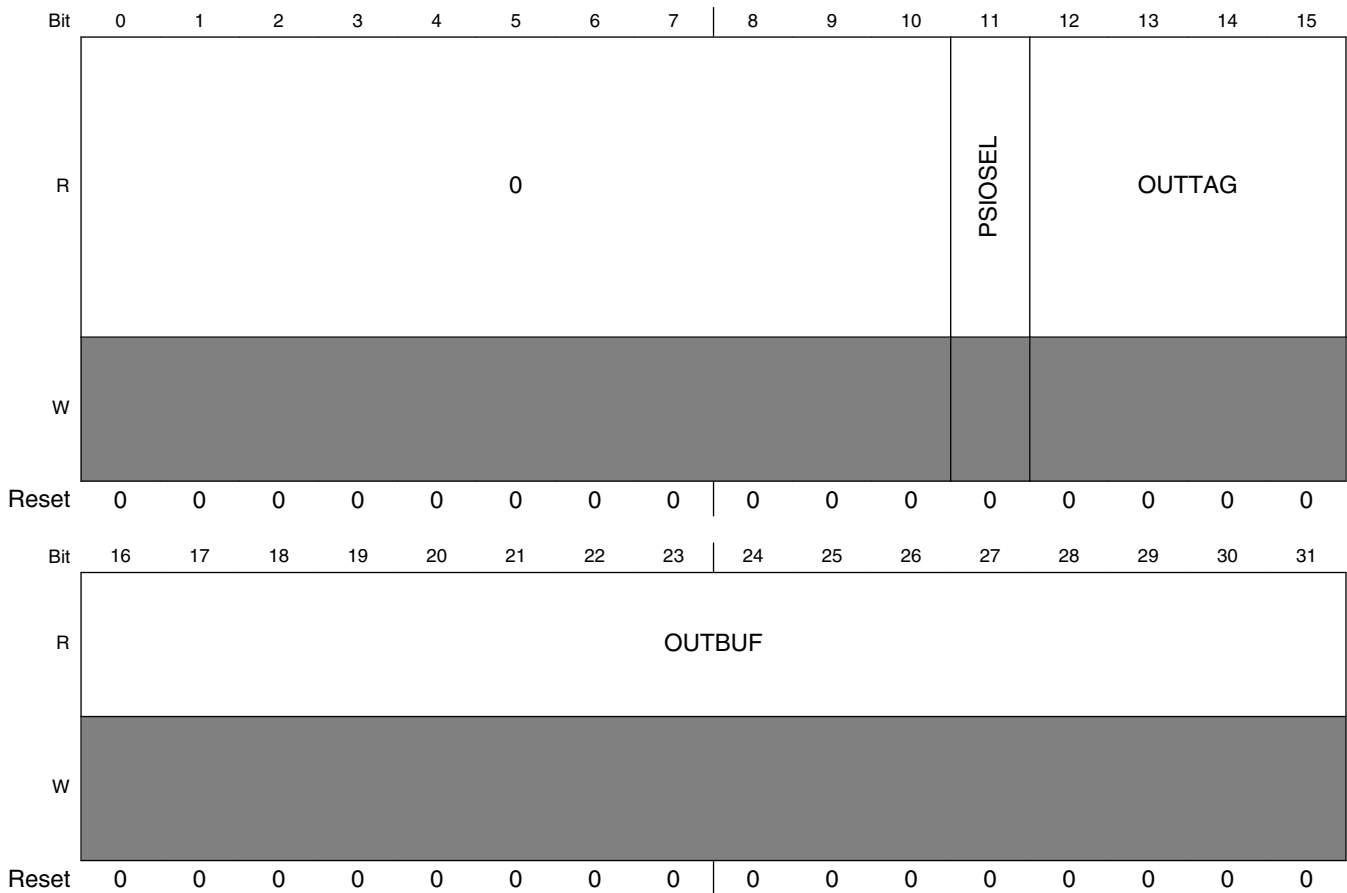
Table continues on the next page...

DECFILTER_IB field descriptions (continued)

Field	Description
14 PREFILL	Decimation Filter Prefill/Filter control bit The PREFILL bit selects the Decimation Filter operation mode. For more details, see Filter Prefill Control Description 0 Decimation Filter normal sample 1 Decimation Filter prefill sample
15 FLUSH	Decimation Filter Flush control bit Assertion of the FLUSH bit initializes the Decimation Filter to a initial state, as defined in Flush Command Description. This bit is self negated and it is cleared only when the data is read and the flush is executed. 0 no flush request 1 flush request
16–31 INPBUF	Input Buffer Data The INPBUF[15:0] bit field carries the sample data to be filtered. This data buffer can be written from the two PSI slave-bus interfaces or by the device slave-bus interface. See Input Buffer Description for more details.

31.4.6 Interface Output Buffer (DECFILTER_OB)

Address: 0h base + 14h offset = 14h



DECFILTER_OB field descriptions

Field	Description
0–10 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
11 PSIOSEL	Selected PSI Output The PSIOSEL bit indicates the selected PSI output that sends the filter result back to the master block. This field has no meaning when in standalone mode or PSI input mixed mode. For more details, see Return PSI (RPSI) and PSI Output Selector (PSIOSEL) . 0 PSI0 is selected to send result data 1 PSI1 is selected to send result data
12–15 OUTTAG	Decimation filter output tag bits The OUTTAG[3:0] bit field is defined as a selector signal and it is used to identify different destinations for the OUTBUF[15:0] data. When an eQADC is the PSI block master, it holds the same number used to address the destination RFIFO.

Table continues on the next page...

DECFILTER_OB field descriptions (continued)

Field	Description
16–31 OUTBUF	Output Buffer Data The OUTPBUF[15:0] bit field is the result data in the decimation filter Output Buffer. It represents a fixed point signed number in two's complement format and is updated only when a decimated result is ready to be transmitted, meaning it contains the last decimated result from the filter.

31.4.7 Filter Coefficient (DECFILTER_COEFn)

Address: 0h base + 20h offset + (4d × i), where i=0d to 8d

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	COEFnSIGNR								COEFn																							
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	

DECFILTER_COEFn field descriptions

Field	Description
0–7 COEFnSIGNR	COEFn sign repeated Bit COEFn[23] is repeated 8 times.
8–31 COEFn	Coefficient n field The COEFn[23:0] bit fields are the digital filter coefficients registers. The coefficients are fractional signed values in two's complement format, in the range (Coef is greater than or equal to -1 and less than 1). NOTE: Reads to this register are sign-extended, meaning the coefficient's sign bit is copied to all 8 most significant register bits. NOTE: Writing to these fields when BSY=1 is not allowed.

31.4.8 Filter TAP (DECFILTER_TAPn)

The TAP register stores, on each filter node, the input sample data and, for the IIR type, the filter intermediary results.

Address: 0h base + 78h offset + (4d × i), where i=0d to 7d

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	TAPnSIGNR								TAPn																							
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	

DECFILTER_TAPn field descriptions

Field	Description
0–7 TAPnSIGNR	TAPn sign repeated Bit TAPn[23] is repeated 8 times.
8–31 TAPn	TAPn Register The read-only TAPn[23:0] bit fields shows the contents of the digital filter tap registers, as fractional signed values in two's complement format, in the range (Coef is greater than or equal to -1 and less than 1). NOTE: Reads to this register are sign-extended, meaning the coefficient's sign bit is copied to all 8 most significant register bits. The content of these registers is meaningless when BSY=1.

31.4.9 Enhanced Debug Input Data (DECFILTER_EDID)

The Enhanced Debug Input Data Register provides read-only access to the sample data received by the decimation filter when the input is selected from the device slave-bus (ISEL=0), allowing the monitoring of filter operation. See [Enhanced Debug Monitor Description](#) for more details. Writes to this register are not allowed.

Address: 0h base + D0h offset = D0h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0																SAMP_DATA															
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	

DECFILTER_EDID field descriptions

Field	Description
0–15 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
16–31 SAMP_DATA	Conversion Sample Data The SAMP_DATA[15:0] bit field carries the data that was loaded in the decimation filter to be processed by the FIR/IIR sub-block. This conversion data is supplied by any of the two PSI slave-bus interfaces and it is only defined when ISEL=0. See Interrupts Requests Description and DMA Requests Description for more details.

31.4.10 Final Integer Value Register (DECFILTER_FINTVAL)

Address: 0h base + E0h offset = E0h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	SUM_VALUE																															
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	

DECFILTER_FINTVAL field descriptions

Field	Description
0–31 SUM_VALUE	Integration Sum Value The SUM_VALUE[31:0] field holds the sum of filtered output values. The 17 most significant bits hold the integer part, and the 15 least significant ones the fractional part of the integration value. The control of the integration sum and update of this register is determined by the register DECFILTER_MXCR. The register is updated only upon an integration output request. SUM_VALUE should be taken as an unsigned number when the integrator is configured for absolute operation (DECFILTER_MXCR bit SSIG=0), and a two's complement signed number otherwise. NOTE: If the DECFILTER_MXCR bit SSAT=1, the integration sum is saturated, so that if the accumulation overflows DECFILTER_FINTVAL holds the value 0xFFFFFFFF for absolute integration (SSIG=0), or values 0x7FFFFFFF (positive saturation) and 0x80000000 (negative saturation) for signaled integration (SSIG=1). If SSAT=0, DECFILTER_FINTVAL holds the integration sum modulo 2^{17} (considering the 15-bit fractional part).

31.4.11 Final Integer Count Register (DECFILTER_FINTCNT)

Address: 0h base + E4h offset = E4h

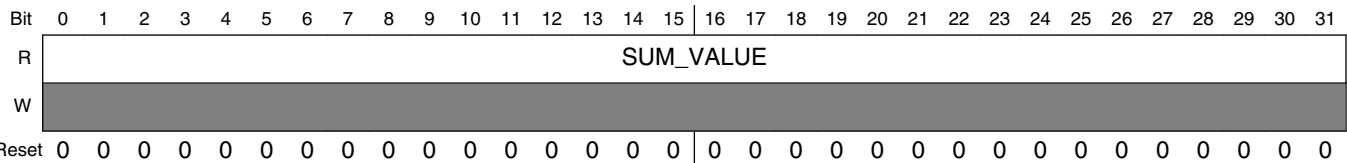
Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	COUNT																															
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	

DECFILTER_FINTCNT field descriptions

Field	Description
0–31 COUNT	Integration Count Value The COUNT field holds the count of filtered outputs integrated. The control of the integration sum and update of this register is determined by the register DECFILTER_MXCR. The register is updated together with DECFILTER_FINTVAL, only upon an integration output request.

31.4.12 Current Integer Value Register (DECFILTER_CINTVAL)

Address: 0h base + E8h offset = E8h

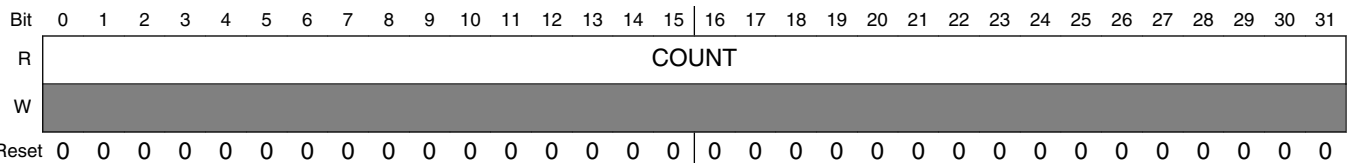


DECFILTER_CINTVAL field descriptions

Field	Description
0–31 SUM_VALUE	Integration Sum Value The SUM_VALUE[31:0] field holds an unsigned number representing the sum of filtered output values, continuously updated as the integration proceeds. The control of the integration sum is determined by the register DECFILTER_MXCR. NOTE: If the DECFILTER_MXCR bit SSAT=1, the integration sum is saturated, so that if the accumulation overflows DECFILTER_CINTVAL holds the value 0xFFFFFFFF for absolute integration (SSIG=0), or values 0x7FFFFFFF (positive saturation) and 0x80000000 (negative saturation) for signaled integration (SSIG=1). If SSAT=0, DECFILTER_FINTVAL holds the integration sum modulo 2¹⁷ (considering the 15-bit fractional part). NOTE: A read on this register automatically commands an update of the register DECFILTER_CINTCNT.

31.4.13 Current Integer Count Register (DECFILTER_CINTCNT)

Address: 0h base + ECh offset = ECh



DECFILTER_CINTCNT field descriptions

Field	Description
0–31 COUNT	Integration Count Value The COUNT field holds the count of filtered outputs integrated. The value is updated only when register DECFILTER_CINTVAL is read, to keep the coherency between the integration and count values.

31.5 Functional Description

31.5.1 Overview

Figure 31-2 shows the block diagram of the Decimation Filter. The Control Logic provides the control signals for all other sub-modules. The PSI data interface is subdivided into two sub-modules, transmitter and receiver, that are accessed by two PSI slave-bus interfaces. The bypass path is used when the filter is disabled and the incoming data can be transmitted back to the master block without being processed by the Filter algorithm. The Filter hardware is implemented in such a way that an IIR (1 x 4 poles) or FIR filter type can be implemented. The selection between the two types of filter algorithms is implemented by the Control Logic sub-block.

The Coefficient register file provides the digital filter coefficients. This block is a register bank with read/write access by the device slave-bus interface. The Filter TAP registers are also accessed through the device slave-bus line interface, providing additional debug capabilities to the Decimation Filter block. The MAC (Multiply Accumulate) sub-block executes the filter arithmetic operations controlled by the Control Logic. The MAC results are routed to the Filter TAP registers and to the output buffer when the result is a decimated filter sample.

31.5.2 Parallel Side Interface (PSI) sub-block description

This section describes the operation of the Parallel Side Interface (PSI) sub-block which is responsible for communication and data exchange between two master blocks (for instance, two eQADC blocks) and the Decimation Filter block. It is important to note that there are two PSI lines (PSI0 and PSI1) but there are only one internal input buffer and only one internal output buffer.

The decimation filter receives sample data from up to two master blocks. The sample data arrives along with the control bits. These control bits are decoded and the proper action is decided in the Control Logic sub-block. When the decimation filter finishes its processing and a result is available, the read request signal is issued to the selected master block. This data transfer request remains set until the result is read by the master block.

When using two or more decimation filter blocks in the device, the output of the second block is connected to the input data of the next block, and the output of the first block is connected to the read data input of the corresponding PSI master block.

31.5.3 Input Buffer Description

The decimation filter receives data samples for filtering from two master blocks (e.g. two eQADCs) using the dual PSI interface, or from the CPU using the device slave bus interface. The data source is selected by the ISEL bit of the module configuration register DECFILTER_MCR. When ISEL selects the PSI interfaces, the user can send data from any of the two master blocks because both are enabled. However, it is necessary to consider the characteristics of the PSI interface itself as described below.

When the device slave bus interface is selected and DMA operation is chosen (DECFILTER_MCR bit DSEL=1), the input data request signal is asserted when the input buffer is empty. When DMA operation is not chosen (DSEL=0) in standalone or PSI output mixed modes, the logic asserts an input interrupt request and the input buffer waits for data from the device slave bus interface.

Input buffer filling is flagged by the DECFILTER_MSR bit IDF. The IDF flag remains set, even after the input data has been consumed and the buffer is free, until it is cleared by software.

Input buffer overrun is detected and flagged by the DECFILTER_MSR bit IVR. The idle (BSY=0) Decimation Filter is able to receive two consecutive input writes without input overrun. For more details, see [Input Buffer Overrun](#).

Note

There is no buffer / control to separate or select data from each PSI interface. The user should know which PSI is being served by the filter module to avoid undesired overrun caused by the two PSI inputs.

When the selected input source is any PSI master block, the PSI master may send timestamp data after related sample data for filtering. The timestamp is sent back to the PSI following the respective filter output, using the same request mechanism. As the decimation filter takes several clock cycles to process a sample, the timestamp is copied into an internal timestamp register until the filtered output is sent out, therefore freeing the input buffer for yet another sample data.

When the input buffer is loaded with a sample data coming from any PSI interface and enhanced debug monitor enabled is enabled (DECFILTER_MCR bits ISEL=0, EDME=1), input DMA or interrupt requests are asserted, so that the PSI input samples can be monitored from the device Sky-Blueslave-bus interface. These interrupt or DMA requests result in input read accesses from the DECFILTER_IB register, unlike write

accesses needed when the same requests are made and the device slave-bus interface is selected for input (ISEL=1). Only the sample data input (not timestamps), can be monitored in this way. See also [Enhanced Debug Monitor Description](#).

Soft reset clears the input buffer indication flags and any data write/read request that was generated in any mode.

31.5.3.1 Input Buffer Overrun

An input overrun occurs when the input buffer is holding input data and new data is received by the filter. See [Input Buffer Description](#) for details of the input buffer.

The input buffer overrun can occur only when the input is enabled (DECFILTER_MCR bit IDIS=0) , and in the following cases:

- When the input buffer has sample data to be processed but the filter is busy and another input (data or timestamp) is received.
- When the input buffer has a timestamp, the internal timestamp register is loaded and the next input data is received.
- When the two PSI interfaces try to write the input buffer at the same time.

As an example of the input data sequence, assume that the filter is enabled and not busy, and all registers are empty. Then a word of sample data is received followed by a timestamp and another word of sample data. No input overrun occurs in this case, because the first sample is immediately transferred to the tap input register, the timestamp is immediately transferred to the internal timestamp storage register, and the second sample can be held in the input buffer until the end of the processing of the sample data by the filter. The input overrun may occur if more input is received before the end of the processing, or if the filter is busy at the beginning of the received sequence.

When the filter is in bypass/disable mode (DECFILTER_MCR field FTYPE=00), the data from the input buffer is transferred to the output buffer, if it is not already full. If the output buffer is full, the input buffer is loaded, and another word of input data is sent, then an input overrun occurs.

Note

Configuring ISEL=1, MIXM=1 and FTYPE=00 (bypass), writes to the DECFILTER_IB are routed directly to the PSI output.

31.5.4 Output Buffer Description

The decimation filter has an output buffer to send filtering results to a master block using any PSI, or to the CPU using the device slave-bus, as selected through the `DECFILTER_MCR` bits `ISEL` and `MIXM` bits.

Note

There is only one output buffer to serve the two PSI outputs. The PSI output selection is controlled by the PSI input field `RPSI` or by the bit field `PSIOSEL` in the `DEC_FILTER_IB` register.

Filtering of prefill inputs do not update the output buffer, so the flag `ODF` is not set.

When filter types `IIR` and `FIR` are selected and the input source is any PSI master block (normal or PSI input mixed modes), the output buffer receives data from the MAC sub-block or from the timestamp storage register. The result from the MAC is written immediately after the processing if the decimation is enabled. However, the timestamp data is enabled to be written in the output buffer only when the output buffer is empty, the decimation count is reached, and the corresponding data was also ready to be transmitted. When a new word of data is available in the output buffer, a read request signal is sent to a master block. The master block has to send the corresponding read commands to clear the read request signal. In this configuration, the core can always read the output buffer. The flag `ODF` is set when the buffer is updated.

When in filter operation mode with input from the device slave-bus (`ISEL=1`), only sample data is processed by the filter, as there is no way to input timestamps. When the filter result from the MAC is ready, this is immediately written to the output buffer when the decimation count is reached. The flag `ODF` is set when the buffer is updated. It also generates a DMA read request if `DSEL=1` in standalone or PSI input mixed modes .

When the filter is bypassed (`FTYPE=00`), and a PSI is selected as output (normal mode or PSI output mixed mode), the data written into the input buffer waits until the output buffer is empty before passing the data. This is needed because the master block takes some clock cycles to send the read commands to the decimation filter after the read request signal is asserted. When the filter is bypassed and the device slave-bus is selected as output (standalone mode or PSI input mixed mode), the data written into the input buffer is immediately written into the output buffer. The flag `ODF` is set when the buffer is updated and a DMA read request is asserted if `DSEL=1`.

Soft reset clears the output buffer, as well as any data read request generated in any mode.

31.5.4.1 Output Buffer Overrun

An output overrun occurs when the output buffer is holding output data (sample or timestamp) that has not been read is overwritten with another word of data (sample or timestamp).

Output overruns are flagged by the DECFILTER_MSR bit OVR, if the output buffer is updated when not empty. The output buffer empty condition depends on the mode and output selection as follows:

- If PSI0 or PSI1 is selected as output (normal mode or PSI output mixed mode), the output buffer is considered empty when the last output has been read.
- If cascade mode is selected, the output buffer is considered empty if the last output request was acknowledged.
- If the device slave-bus is selected (standalone or PSI input mixed mode) for output and DMA is selected (DSEL=1), the output buffer is considered empty if the last output has been read.
- If the device slave-bus is selected (standalone or PSI input mixed mode) for output and DMA is not selected (DSEL=0), the output buffer is considered empty when the ODF flag is negated.

Note

When the device slave-bus is selected (standalone or PSI input mixed mode) for output and DMA is not selected (DSEL=0), the ODF flag must be cleared to avoid overrun, even if its corresponding interrupt is not used (ODEN=0).

Note

The normal use is to receive the data stream from one specific PSI and transmit the results using one specific PSI. Input and output timing care are necessary to avoid output overrun when output results are sent to both PSI alternately.

Prefill inputs do not cause IIR or FIR output overrun in standalone, normal, and cascade tail modes, but can cause overruns in cascade head or middle configurations (see [Cascade mode](#)).

When bypass is selected, the output overrun does not occur because the data written into the input buffer is written into the output buffer only when this buffer is empty, but an input overrun may still occur (see [Input Buffer Overrun](#)).

31.5.4.2 Triggered Output Result Description

The posting of a filter output, either to a PSI interface or to the device slave-bus can be controlled by an additional input trigger signal (see [Decimation trigger signal](#)). It allows the decimation to be controlled by another circuit, instead of the internal decimation counter.

Triggered output operation is enabled by the bit TORE in the configuration register DECFILTER_MCR. When triggered output is enabled, the decimation count configured by DECFILTER_MCR field DEC_COUNTER is ignored. The decimation filter detects the rising edge or the falling edge of the trigger signal as selected by the configuration field TMODE in the DECFILTER_MCR register. When the corresponding edge is detected, the output buffer is enabled to receive the next filter result. This means if a filtering process is occurring, the output buffer read request is only set when the filtering ends and a new result is available.

Note

This mode is not affected by the prefill sample indication then a result is pulled into the output buffer when TORE is enabled and a trigger is detected. Therefore, the initialization phase of the digital filter need to be controlled by the timing of asserting the TORE bit or by discarding the initial sampled results.

Note

The time interval between two consecutive trigger events should be greater than 60 clock cycles. This is necessary to avoid possible trigger overruns.

The input trigger signal can also be used as a simple filtered output enable: in this trigger mode, when the input trigger signal is asserted, every filtered output is posted to the output buffer; no output is posted when the signal is negated. This trigger mode and the assertion polarity (active 0 or 1) of the input trigger signal is also defined in the DECFILTER_MCR field TMODE.

31.5.5 Bypass Configuration Description

Bypass operation is configured by setting the field FTYPE[1:0] of module configuration register DECFILTER_MCR to 00. In this case, the input sample and tag are sent to the output with no change. This behavior is independent of the ISEL/MIXM setting. The following applies to the bypass configuration:

- flush is ignored
- prefill is ignored
- trigger or counted decimation is ignored
- BSY bit is not set
- the input and output flags are set

Note

Bypass must not be configured in cascade mode (see [Cascade mode description](#)).

31.5.6 IIR and FIR Filter

This section describes the IIR filters implemented in the Decimation Filter block. [Figure 31-33](#) shows the filters functional diagrams. The figure shows a IIR filter of fourth order (4poles). The filter topology, not the hardware, is represented. The hardware implementation is based on a MAC unit controlled by an FSM which implements the filter algorithm.

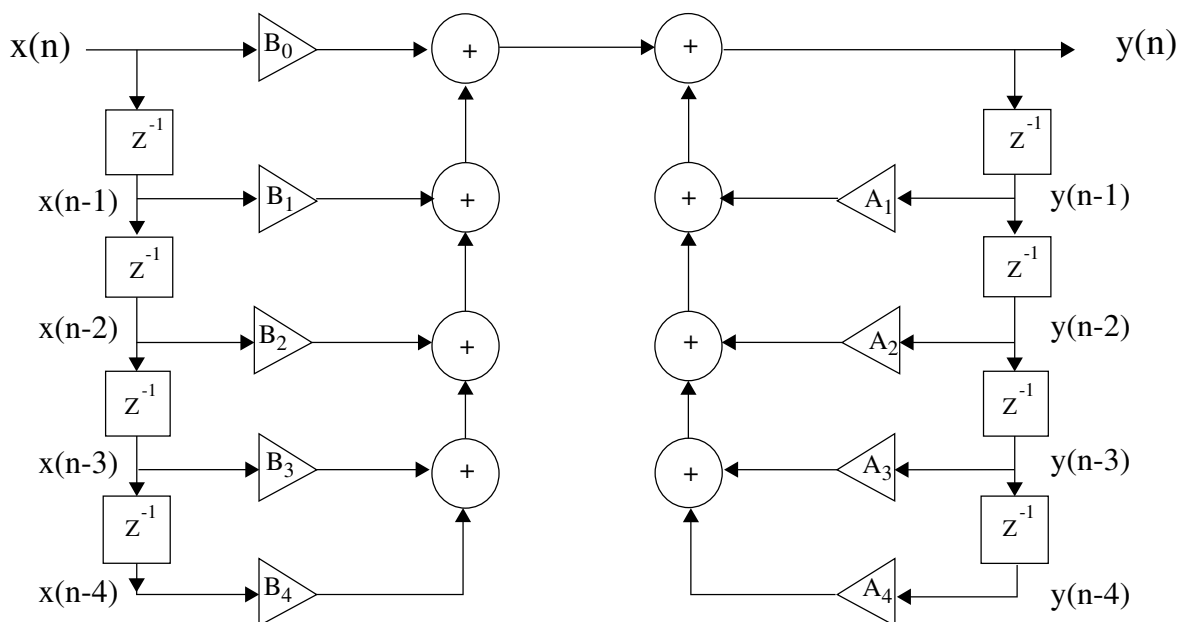


Figure 31-33. 1 x 4 Poles IIR Filter Functional Diagram

The difference equation for the IIR filter of [Figure 31-33](#) can be written as:

$$y(n) = \sum_{i=0}^N B_i x(n-i) + \sum_{j=1}^M A_j y(n-j)$$

Figure 31-34. Eqn-1

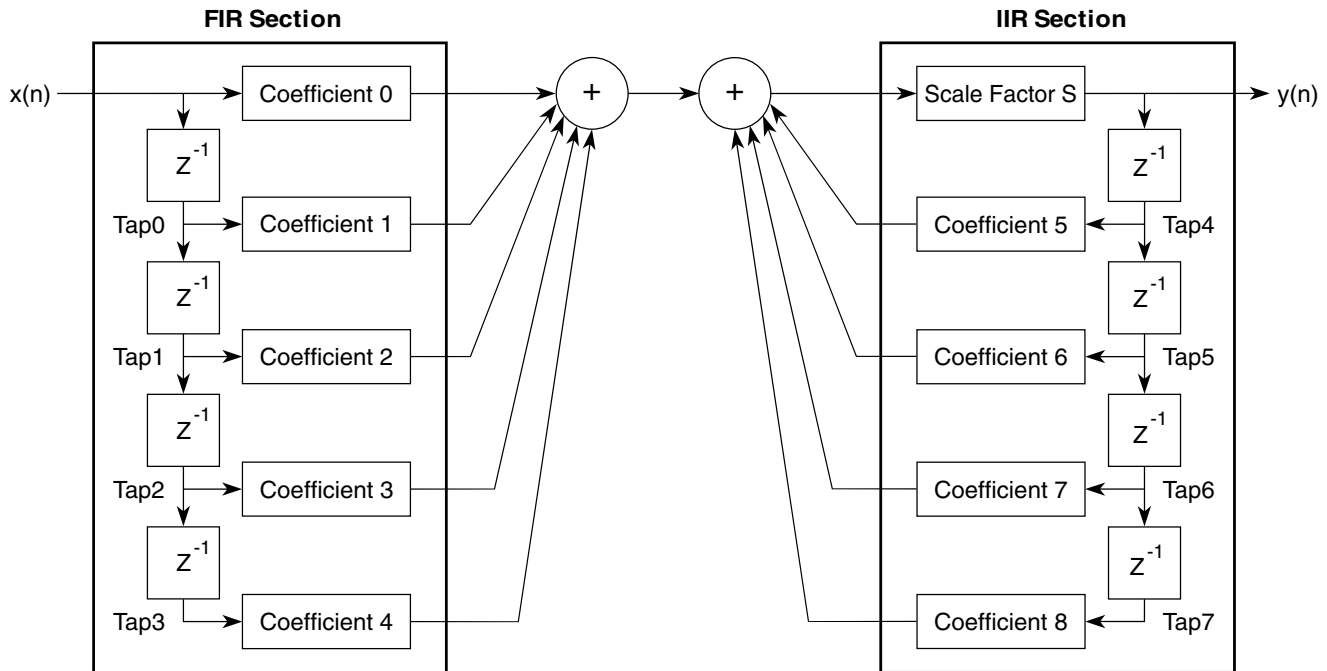
where $x(n)$ is the filter input at time n , $y(n)$ is the filter output at time n , N is the number of feed-forward filter coefficients minus one, B_i are the feed-forward filter coefficients, M is the number of feed-back filter coefficients, and A_j are the feedback filter coefficients.

Figure 31-34 can be written as:

$$y(n) = S \left\{ \sum_{i=0}^N \frac{B_i}{S} x(n-i) + \sum_{j=1}^M \frac{A_j}{S} y(n-j) \right\}$$

Figure 31-35. Eqn-2

Where all the coefficients are scaled down by S . The block diagram for Figure 31-35 is shown in Figure 31-36 in a fourth-order IIR filter implementation where the coefficients A_j and B_i are called *coefficient n* , where $n=0-8$.

**Figure 31-36. Fourth Order IIR Filter Implementation Block Diagram**

The fourth order IIR filter is implemented with a FIR section followed by an IIR section. If the FIR type filter mode is selected, the IIR section is converted into an FIR section. In this case the order of the FIR filter is twice the IIR filter order, since all the TAP and coefficient registers are allocated for the FIR section. The Filter configuration paths are shown in Figure 31-37. Multiplexer A controls the *bypass* filter path and multiplexer B controls/selects the filter mode of operation, to either IIR mode or FIR mode. The

selection is controlled by the FTYPE[1:0] bits in the Filter Module Configuration register. The order of the filter can be controlled by setting the appropriate *filter coefficients* to zero.

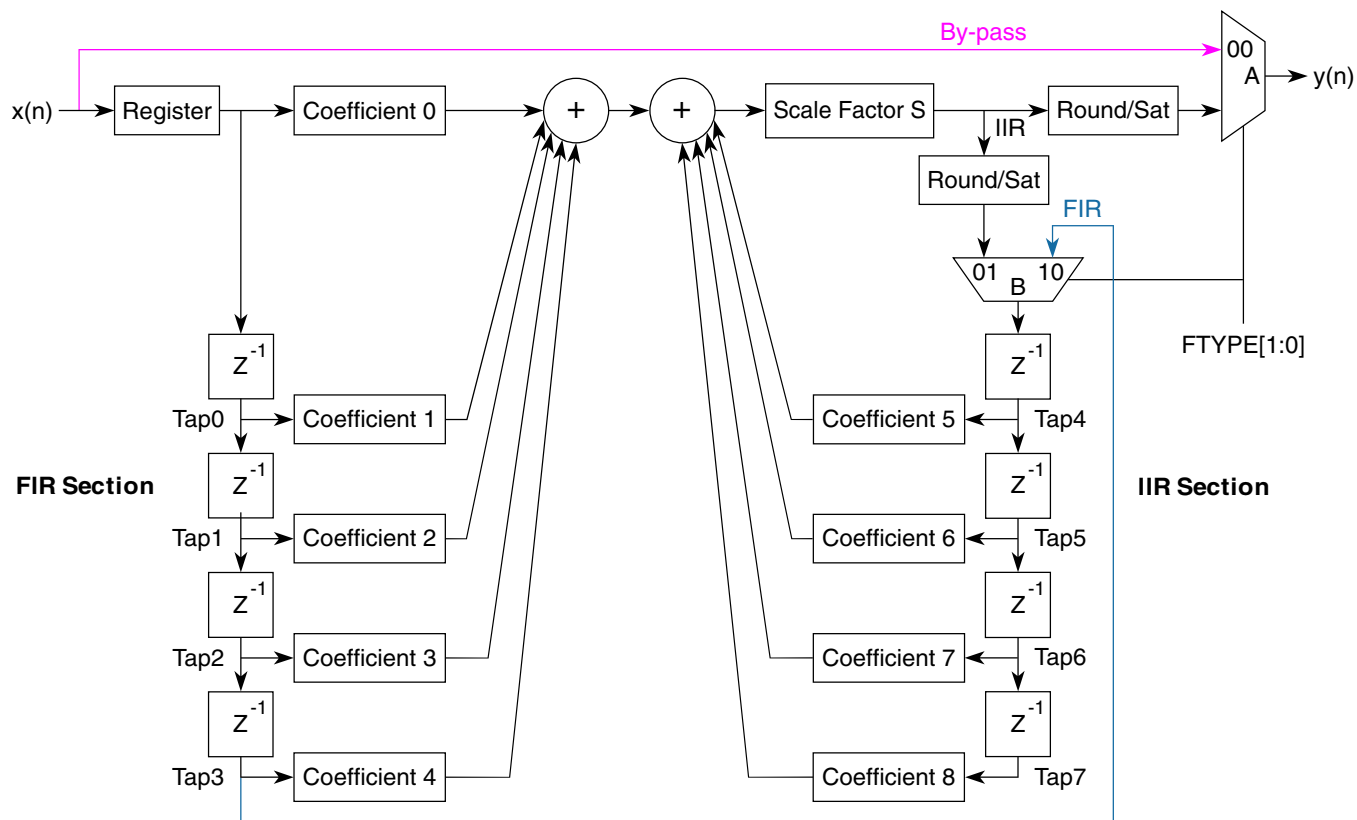


Figure 31-37. Filter Configuration Paths (FIR or 1x4Poles IIR)

31.5.6.1 Rounding

The Decimation Filter performs rounding operations in two different locations, as shown in [Figure 31-37](#) :

- to obtain the filter output result with 16 bits
- to obtain the IIR feedback result to be stored in tap4 registers with 24 bits

The rounding mechanism implements the Convergent Rounding methodology (also known as round-to-nearest even number), which makes the decision on rounding up or down based on the value of the lower portion of data to be rounded (LS_WORD). The rounding up/down condition is equal to the traditional rounding except when the

LS_WORD has the format { 1000...00 }. In this particular case, the rounding procedure is like the example of Figure 31-38. If the MS_WORD is odd, the value is rounded up. Otherwise the value is rounded down.

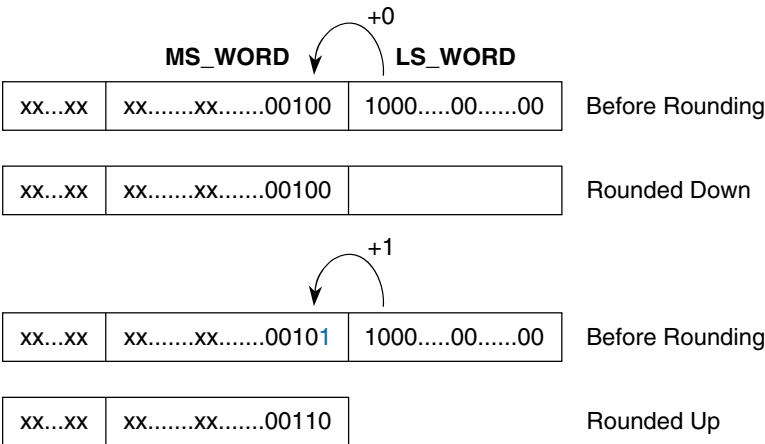


Figure 31-38. Convergent Rounding Methodology

31.5.6.2 Saturation

Filter output saturation occurs when an overflow or underflow condition of the filter is detected by dedicated logic, and if it is enabled by the SAT control bit of the configuration register DECFILTER_MCR. In this condition, the filter output is set to a saturated value equal to the maximum or minimum value that can be represented by the 16-bit output port. Also, for the IIR filter an equivalent logic is used to assert the saturation for the 24-bit feedback result.

31.5.7 Filter Prefill Control Description

A prefill indicates that the input data should be accepted by the Decimation Filter, but no decimated output should be generated while the control field indicates prefill. Therefore the prefill function is used in the beginning of the filter operation to initialize and stabilize the Decimation Filter without generating decimated samples. In addition, the prefill does not operate when the filter is in bypass (FTYPE=0b00).

When ISEL=0, the M_CTRL[1:0] field in the PSI bus data, input and output controls the prefill. When ISEL=1, it is controlled by the PREFILL field in the DECFILTER_IB register. The prefill control is usually activated only in a certain number of words of sample data in the beginning of the input data sequence.

When the prefill control is set, the decimation filter block operates as follows:

- Input data is processed normally by the digital filter and tap values are updated.
- The decimation counter is maintained in reset value.
- The output buffer is not updated and no output interrupt or read request is generated.
- The accompanying timestamp for the identified prefill conversion data is not bypassed.
- The overflow detector/flag operates normally and the error interrupt request is set if enabled.

In Cascade Mode, prefill control is ignored in all cascaded filters except the Tail one: the prefill samples are filtered, decimated and forwarded to the next block with the prefill indication. The Tail block takes the samples with prefill indication, filter them and discard the result.

Note

The combined decimation count of the cascade combo may not be reset after the last prefill sample. Unlike in a single (non-cascaded) filter, the number of non-prefill inputs until an output comes out depends on the number of past prefill inputs.

31.5.8 Timestamp Data Transmission

The timestamp information is identified by the master block using the M_CTRL[1:0] bit field of the PSI bus data, input and output. For timestamp data, the input data and tag values that come with M_CTRL[1:0] bits set to 0b10 are sent back to the master block without changing. However, some additional conditions are considered:

- The timestamp is additional information that accompanies a sample conversion data. The PSI master block sends the decimation filter the conversion data with control bits for either prefill or filter operation. This data may optionally be followed by the corresponding timestamp data. When the corresponding conversion data is marked for prefill, the timestamp data is not sent to the output buffer. This occurs because the filter result is not sent to the output buffer.
- Similarly, when the filter is decimating the results, the timestamp is only sent to the output buffer if the corresponding received conversion data has generated a filter output that is selected by the decimation counter to be sent to the output buffer. Other received timestamps that come with data not selected by the decimator are discarded.

- Sending two consecutive words of timestamp data is not allowed: there must be at least one conversion data between two timestamp inputs. In normal operation, the filter should receive only one timestamp for each word of conversion data to be filtered.
- Timestamps are not allowed in PSI Input Mixed Mode.
- When both PSI are sending data / timestamp to the same decimation filter, the user should control the timing between the two PSIs to avoid loss of information and erroneous mixing of data and timestamp.

31.5.9 Flush Command Description

The flush signal is used by the Decimation Filter to execute a partial reset of the filter. This is useful when the same filter is used on a new set of data samples after finishing the filtering of another set of data.

When the flush control is detected, all filter TAPs are cleared and the DEC_COUNTER[3:0] field in the status register DECFILTER_MSR is reset.

The flush function does not clear the Coefficient registers file in the Decimation Filter, thus it is not required to re-write these registers after a flush. The integrator accumulator and sample count are not cleared either. The output buffer also keeps the last result and may be retrieved until the next output is posted.

The flush control precedes the input data to be filtered. Therefore, the corresponding sample data is processed by the block after the flush. When ISEL=0, the field M_FLUSH in the PSI bus data, input and output is processed. When ISEL=1, the field FLUSH in the DECFILTER_IB register is processed. Note that a word of valid sample data can be available at the same time the flush signal is asserted. In this case the flush is executed and the sample is processed after the flush.

When ISEL=0, flush bit M_FLUSH must not be asserted when the input data is a timestamp (M_CTRL=10).

When the filter is disabled by the FTYPE[1:0] control bit field, the flush command is not executed.

Note

In Cascade Mode, the flush command is forwarded to the next cascaded block together with the output, after the decimation count. Therefore it is possible that, in a given moment, the taps of a cascaded block are zeroed after a flush input, while the following ones still retain the old values.

31.5.10 Soft-Reset Command Description

The Soft-Reset command is requested through the self-negated bit SRES of the DECFILTER_MCR register and provides the CPU with the capability to initialize the Decimation Filter through the slave-bus interface. The procedure below must be performed for a software reset when the filter is active:

1. Disable filter inputs, writing DECFIL_MCR bit IDIS = 1.
2. Poll the register DECFIL_MSR until the bit BSY is 0.
3. Repeat the step 2 polling; this is necessary to cover the case when a sample is left in the input buffer.
4. Write DECFILTER_MCR bit SRES = 1.

After the software reset is issued, all internal Filter TAP registers, the decimation counter, the integrator outputs (except DECFILTER_CINTCNT), and the state machine are put in the initial state. The status register DECFILTER_MSR is also cleared. The Coefficient registers are not affected by the SRES. In case there is some filter processing, the filter process is aborted and the last sample is discarded. In addition, data in the input buffer waiting to be processed, and data in the output buffer waiting to be read, are discarded (the requests of service are cleared). The software reset command has high priority and the BSY bit is set during its operation.

The configuration registers DECFILTER_MXCR and DECFILTER_MCR are also not affected by a soft reset, except the bit SRES that is self-negated and is always read as zero.

When in debug or freeze mode, the soft reset is executed but the filter remains in debug or freeze mode.

Note

It is recommended to clear the IBIE bit before a software reset, especially if ISEL changes, in order to avoid unwanted interrupt requests.

Note

DMA transfers must not be active during soft reset. Data loss can occur.

31.5.11 Interrupts Requests Description

31.5.11.1 Block Interrupt Request

There are several interrupt request events that can be enabled using the module configuration register `DECFILTER_MCR`. Basically, the interrupt request can be issued under any of the following conditions:

- when a word of input data is received
- when a word of output data is available
- when an error has occurred.

The input data flag `IDF` is set when a word of data is received from the CPU when in standalone mode, or when a word of data is received from any PSI when it is not in standalone mode (normal mode). It is not used to generate the read or write requests (as defined in [Input Buffer Interrupt Request](#)) when `DSEL=0`.

Output data is available and its flag (`ODF`) is set when the input data sample is processed by the filter and the decimation counter matches the decimation rate value. It is not used to generate the read requests (as defined in [Output Buffer Interrupt Request](#)) when `DSEL=0`.

An error event in the decimation filter block is defined as one of these events:

- Overflow in the filter, flagged by `OVF`
- Overrun in the decimation filter input, flagged by `IVR`
- Overrun in the decimation filter output, flagged by `OVR`
- Overrun in enhanced debug monitor, flagged by `DIVR`
- Integrator overrun, flagged by `SVR`
- Integrator value exception, flagged by `SSE`
- Integrator count exception, flagged by `SCE`

An overflow occurs when the two's-complement result value from the MAC accumulator is out of the range of values that can be stored in tap register 4 (IIR) or in the output register.

An input overrun occurs when the input buffer is holding a word of input data and one more word of data is received by the filter. See [Input Buffer Overrun](#), for more details.

An output overrun occurs when a new word of data is sent to the output buffer but the previous word of data has not been handled yet. See [Output Buffer Overrun](#), for more details.

These flags can be set by the PSI events, however they are only cleared by:

- the CPU, or
- by the soft reset command in the DEC_FILTER_MCR, or
- by the clear flag registers in the DEC_FILTER_MSR register.

31.5.11.2 Input Buffer Interrupt Request

This interrupt is enabled by the register IBIE in the DECFILTER_MCR register and is asserted only when DSEL=0. This request is flagged in the DECFILTER_MSR by field IBIF.

In standalone and PSI output mixed modes, the input buffer interrupt request is asserted when the input buffer is available to receive a conversion sample and DMA operation is not selected (DSEL=0), meaning the block is requesting data to be written into the input buffer. The interrupt request is cleared when the CPU writes a "1" in field IBIC of the DECFILTER_MSR register, or by the soft reset command.

When in normal, cascade or PSI input mixed modes with enhanced debug enabled (ISEL=0 and EDME=1, DSEL=0), the input sample data can be read by the CPU when this interrupt request is asserted. The interrupt is asserted just when a new word of sample data is supplied to the filter sub-block to be processed. As this filter register is overwritten by the next word of sample data, an input read overrun event can occur (the DIVR bit in DECFILTER_MSR register is asserted) if the interrupt request is not cleared before, or at the same time as, the new sample arrives to set the interrupt. This DIVR bit is cleared by the DIVRC bit in the status register DECFILTER_MSR. However, in enhanced debug monitoring, the set condition has higher priority than the clear. This means that if the set condition and the clear bit IBIC occur at the same time, the interrupt remains asserted.

31.5.11.3 Output Buffer Interrupt Request

This interrupt is enabled by the register OBIE in the DECFILTER_MCR register and is asserted only when DSEL=0. This request is also indicated in the field OBIF of the DECFILTER_MSR register.

When in standalone mode, the output buffer can be read by the CPU with the DMA disabled. The output buffer interrupt request is asserted when the output buffer receives a new result from the filter sub-block. This means the block is requesting data to be read by the CPU.

The output buffer interrupt request can also be asserted due to an integrator result being ready, as flagged by the DECFILTER_MXSR bit SDF, when the DECFILTER_MCR bit SDIE=1. Note that both the filter output and integrator share the same interrupt source.

This interrupt request is cleared when a "1" is written in the bit OBIC and/or bit SDFC of the DECFILTER_MXSR register, or by the soft reset command.

31.5.12 DMA Requests Description

The DMA function for input and output buffers is enabled using DSEL=1 in the DECFILTER_MCR register. The DMA function for integrator result is described in another section.

31.5.12.1 Input Buffer DMA Request

This DMA request is enabled by the ISEL and DSEL bits in the DECFILTER_MCR register.

When in standalone mode, the input buffer can be written by the DMA. The input buffer DMA request is asserted when the input buffer is available to receive a conversion sample (it is not holding a word of data). This DMA request is cleared when an input data word is written in the input buffer. Therefore, the DMA request is always cleared before it is asserted again.

When in normal mode with enhanced debug enabled (ISEL=0 and EDME=1, DSEL=1), the input sample data can be read by the DMA when this DMA request is asserted. The request is asserted when a new word of sample data is written into the input buffer to be processed. As this filter register is overwritten by the next word of sample data, a DMA read overrun event can occur (the DIVR bit in DECFILTER_MSR register is asserted) if

the DMA request is not cleared before, or at the same time as, a new sample arrives to set the DMA request. This DIVR bit is cleared by the DIVRC bit in the status register DECFILTER_MSR or by soft reset.

This DMA request is cleared when the DMA transfer is complete or by soft reset.

31.5.12.2 Output Buffer DMA Request

This DMA request is enabled by the ISEL, MIXM and DSEL bits in the DECFILTER_MCR register.

When in standalone mode, the output buffer can be read using the DMA. The output buffer DMA request is asserted when the output buffer receives a new result from the internal filter sub-block. This DMA request is cleared when the output buffer is read by the processor.

The output buffer DMA request is also asserted when an integrator output is ready, if DECFILTER_MXCR bit SDMAE=1 and the filter output is directed to a PSI interface (ISEL=MIXM).

The DMA request is also cleared by soft reset.

31.5.13 Freeze Mode Description

The freeze mode operation is asserted using the FREN enable bit and FRZ bit in the DEC_FILTER_MCR register or by the modules debug input .

It is not possible to enter freeze mode when the module is disabled by the configuration bit MDIS.

In case of a freeze mode request during the processing of an input sample, the current processing is finished and then the module enters freeze mode.

Access to input and output buffers remain operational in freeze, as well as their related flags.

Note

For cascaded blocks (see [Cascade mode description](#)), it is recommended to freeze only the Head block. If it is really necessary to freeze all the blocks, it is recommended that they

are frozen sequentially starting from the Head block towards the Tail one. Unfreezing is recommended to be done sequentially from the Tail block towards the Head one.

31.5.14 Enhanced Debug Monitor Description

This feature is enabled by the EDME bit in the configuration register DECFILTER_MCR. The monitoring operation works either in normal or cascade modes, when the sample data is supplied by a master block through the PSI slave-bus interfaces (ISEL=0) or another cascaded block (see [Cascade mode description](#)). The Enhanced Debug Monitor feature makes the input sample data also available in the DECFILTER_EDID register. A DMA or interrupt request (selected by DSEL) indicates a new input was fed and DECFILTER_EDID was updated. The input is processed normally by the filter.

An Enhanced Debug Input Data Register (DECFILTER_EDID) overrun can occur if a sample is not read by the CPU or DMA before overwritten by a new sample. The overrun is indicated in a separate flag DIVR in the status register DECFILTER_MSR. If the ERREN bit is set in the DECFILTER_MCR configuration register, this overrun asserts the module interrupt request.

31.5.15 Integrator

The hardware sample integrator accumulates the filter output values for determined periods.

31.5.15.1 Integrator Inputs

The integrator can be fed either by raw or decimated filter outputs, selected by the DECFILTER_MXCR bit SISEL (see [Module Extended Config Register \(DECFILTER_MXCR\)](#)). The accumulated input value taken can be the filtered sample "as is" (signaled), or its absolute value, depending on the DECFILTER_MXCR register bit SSIG.

Note

The integrator accumulates input samples when bypass is selected.

31.5.15.2 Integrator Outputs

The integrator output is either:

- the 32-bit, fixed point unsigned accumulation of the absolute values from the filter output, when the integrator is configured for absolute operation (DECFILTER_MXCR bit SSIG=0). This resolution allows a minimum of 131071 samples to be integrated before an overflow occurs in absolute operation, or 65536 samples in signed operation.
- the 32-bit, fixed point signed two's complement accumulation of the signed values from the filter output, when the integrator is configured for signed operation (SSIG=1).

The fractional part of the accumulation is 15 bits wide in both cases.

An accumulation overflow is flagged by the DECFILTER_MXSR bit SSOVF. The accumulator can overflow in either of the ways below, selected through the DECFILTER_MXCR bit SSAT:

- saturated accumulation (SSAT=1), so that an overflow results in the value of 0xFFFFFFFF for absolute value accumulation (SSIG=0), or 0x7FFFFFFF (positive) and 0x80000000 (negative) for signaled accumulation (SSIG=1).
- non-saturated accumulation (SSAT=0), so that an overflow results in the modulo 2^{17} accumulation value. This operation is only allowed in absolute accumulation (SSIG=0).

The integrator output value becomes available in register DECFILTER_FINTVAL (see [Final Integer Value Register \(DECFILTER_FINTVAL\)](#)) when an integrator output request is issued. The integrator output request can be issued in the following ways:

- by hardware, controlled by an external signal; the enabling and the selection of the signal request modes is done through the DECFILTER_MXCR field SRQSEL (see [Module Extended Config Register \(DECFILTER_MXCR\)](#));
- by software, writing 1 to the DECFILTER_MXCR bit SRQ;

The SSOVF flag is asserted upon an integrator output request, based on the overflow state of the internal accumulator. This internal overflow state is cleared upon an output request, just after SSOVF is asserted, or upon an integrator reset (see [Integrator Reset](#)). The internal overflow state is also cleared by writing SSOVFC to 1, but only in saturated accumulation. Therefore, a non-saturated overflow that occurs before an SSOVF clear is still flagged in the next output request.

The integrator output request also updates the register `DECFILTER_FINTCNT`, which holds the number of samples accumulated into the register `DECFILTER_FINTVAL`. This internal accumulated sample counter can operate either in a saturated or "wrapped" count mode, as selected by the `DECFILTER_MXCR` bit `SCSAT`. In both cases, the counter overflow is flagged by the `DECFILTER_MXSR` bit `SCOVF`.

The `SCOVF` flag is asserted upon an integrator output request, based on the overflow state of the internal counter. This internal overflow state is negated upon an output request, just after `SCOVF` is asserted, or upon an integrator reset (see [Integrator Reset](#)). The internal overflow state is also negated by writing `SCOVFC` to 1, but only in saturated count. Therefore, a non-saturated overflow that occurs before an `SCOVF` clear is still flagged in the next output request.

An integrator output update can also issue a DMA or interrupt request. The interrupt and DMA requests are the same ones used for the filter output buffer (see [Output Buffer Interrupt Request](#) and [Output Buffer DMA Request](#)). The `DECFILTER_MCR` bit `SDIE` is used to enable integrator interrupts, and the `DECFILTER_MXCR` bit `SDMAE` enables the DMA integrator requests. The integrator DMA request uses the same signal as the filter output DMA request, so one must never use any configuration that allows both the integrator and filter output to make DMA requests.

Integrator output updates are flagged by the `DECFILTER_MXSR` bit `SDF`. The integrator overrun is detected in the same way as a filter output buffer overrun, and is flagged by `DECFILTER_MXSR` bit `SVR`. An integrator overrun also generates an error interrupt if the `DECFILTER_MCR` bit `ERREN`=1 (see [Output Buffer Overrun](#)).

Registers `DECFILTER_CINTVAL` and `DECFILTER_CINTCNT` provide a way to poll intermediate integration values and sample counts, respectively (see [Current Integer Value Register \(DECFILTER_CINTVAL\)](#) and [Current Integer Count Register \(DECFILTER_CINTCNT\)](#)). `DECFILTER_CINTVAL` is updated whenever the integrator is reset or a new sample is accumulated. `DECFILTER_CINTCNT` is updated only when `DECFILTER_CINTVAL` is read, so that coherency between the value and count values is guaranteed. Therefore, the read access order of that pair of registers must be `DECFILTER_CINTVAL` first, followed by `DECFILTER_CINTCNT`.

Note

The flags `SSOVF` and `SCOVF` can also be asserted when `DECFILTER_CINTVAL` is read. The `SSOVF` and `SCOVF` set and clearing rules apply for the `DECFILTER_CINTVAL` read the same way as for an integrator output request.

31.5.15.3 Integrator Reset

The integration value is reset to the value of zero, in the following ways:

- by hardware: on hardware reset, or controlled by an external signal; the enabling and the selection of the zero signal modes is done through the DECFILTER_MXCR field SZROSEL (see [Module Extended Config Register \(DECFILTER_MXCR\)](#));
- by software: on software reset, or writing 1 to the DECFILTER_MXCR bit SZRO;

The integrator reset also zeroes the internal counter of accumulated samples and the internal overflow state (but not SSOVF and SCOVF). Software and hardware resets reset all integrator registers immediately. Integrator zero command from external signal or by software (SZRO) affects the integrator registers and flags as follows:

- DECFILTER_CINTVAL resets immediately;
- DECFILTER_CINTCNT does not reset immediately; it is updated only upon a DECFILTER_CINTVAL read, loaded with the number of integrated samples occurred after the reset;
- DECFILTER_FINTVAL and DECFILTER_FINTCNT do not reset immediately; being updated only upon a new output request (see [Integrator Outputs](#)); if a integrator software zero command (through SZRO bit) and an integrator output request (through SRQ bit) are made at the same time, the registers DECFILTER_FINTVAL and DECFILTER_FINTCNT are updated with the last internal values before reset; the same applies to simultaneous integrator zero command and output request by hardware signal;
- the SSOVF and SCOVF flags do not negate; however, the internal overflow states which assert SSOVF and SCOVF do reset immediately, so that the next output update (either by hardware request, software request or DECFILTER_CINTVAL read) before an overflow does not assert SSOVF/SCOVF.

Note

The integrator reset does not depend on the integrator enabling (see [Integrator Enabling and Halting](#)).

31.5.15.4 Integrator Enabling and Halting

Two mechanisms, enabling and halting, drive the integrator accumulation, allowing it to be controlled by a combination of two distinct sources, both software, both hardware, or one hardware and other software. Values are accumulated when the integrator is enabled and not halted. The integrator halt and enable states can be controlled in the following ways:

- by hardware, through external signals ; the enabling and the selection of the signal request modes is done through the DECFILTER_MXCR fields SENSEL and SHLTSEL, respectively (see [Module Extended Config Register \(DECFILTER_MXCR\)](#));
- by software, through the same DECFILTER_MXCR fields SENSEL and SHLTSEL. Note that these fields are in different bytes, so that two distinct, concurrent software tasks can avoid coherency problems by changing the fields using byte read-modify-write accesses.

Note

Enabling and halting does not affect output requests or integrator reset.

31.5.15.5 Integrator Exceptions

Integrator may run into exception states due to overflow, either of the accumulated value or the sample counter. Exceptions are flagged by the DECFILTER_MXSR bits SSE, for sum value exception, and SCE, for counter exception. These flags generate an error interrupt, if it is enabled (see [Block Interrupt Request](#)).

The accumulator exception condition depends on whether it operates in saturated mode or not, as follows:

- In Saturated operation (DECFILTER_MXCR bit SSAT=1): a sum exception occurs (SSE=1) whenever an overflow is flagged; SSE asserts together with SSOVF.

- In Non-saturated operation (DECFILTER_MXCR bit SSAT=0): a sum exception occurs (SSE=1) when an overflow is flagged and the DECFILTER_MXSR bit SSOVF is already set to 1.
- In Non-saturated operation, an accumulator exception also occurs if the accumulator overflows twice without any update of the final integrator value DECFILTER_FINTVAL or the current integrator counter DECFILTER_CINTCNT (by a read to the DECFILTER_CINTVAL register), neither an integrator reset occurs. The SSOVF flag does not assert in this situation.

Note

The SSOVF flag can only be asserted upon a hardware request, a software request, or when DECFILTER_CINTVAL is read, based on the internal accumulator overflow state.

Similarly, the sample counter exception condition depends on whether it operates in saturated mode or not, as follows:

- In Saturated operation (DECFILTER_MXCR bit SCSAT=1): a counter exception occurs (SCE=1) whenever an overflow is flagged; SCE asserts together with SCOVF.
- In Non-saturated operation (DECFILTER_MXCR bit SCSAT=0): a counter exception occurs (SCE=1) when an overflow is flagged and the DECFILTER_MXSR bit SCOVF is already set to 1.
- In Non-saturated operation, a counter exception also occurs if the counter overflows twice without any update of the final count DECFILTER_FINTCNT or the current integrator counter DECFILTER_CINTCNT (by a read to the DECFILTER_CINTVAL register), neither an integration reset occurs. The SCOVF flag does not assert in this situation.

Note

The SCOVF flag can only be asserted upon a hardware request, a software request, or when DECFILTER_CINTVAL is read (also updating DECFILTER_CINTCNT), based on the internal counter overflow state.

31.5.16 Cascade mode description

The cascade mode is defined as a configuration mode of the decimation filter to work together with other ones in a chain arrangement. All blocks in the arrangement, hereafter called a **cascade combo**, are configured to operate in cascade mode by the CASCD[1:0] field in the DECFILTER_MCR register. [Figure 31-39](#) shows an example of cascade combo:

- The figure shows PSI being used for both data input and output, but cascade can also work in standalone or mixed modes.
- The left most block, named Head receives the raw data to be filtered from a PSI master block (or from the device slave-bus interface in standalone/PSI output modes). The head type is configured using CASCD[1:0] = 01 in the configuration register DECFILTER_MCR.
- The right most block, named Tail, is the last filter block in the chain. It sends the output result back to a PSI master block (or to the device slave-bus interface in standalone/PSI output modes). This type of cascaded block is configured using CASCD[1:0] = 10.
- The blocks in between, named Middle, do not exchange data (receive / transmit) with a PSI masterthe device slave-bus, only with other decimation filter blocks. This type of block is configured by setting CASCD[1:0] = 11. Middle blocks are optional in a cascaded combo: two blocks, one Head block feeding a Tail one, can be used in cascade.

Note

The values passed between cascaded blocks can be monitored using Enhanced Debug Monitor (see [Enhanced Debug Monitor Description](#)).

- Each decimation filter block has one cascade-in and one cascade-out ports, besides the PSI connections used in normal mode. The arrows show how they are physically connected. The bold arrows show the connections used in cascade mode.
- The block configurations as Head, Tail or Middle must respect their physical connections such that all the following apply:
 - a Head block feeds a Middle or Tail one
 - a Middle block feeds another Middle block or a Tail one
 - Tail feeds no other block, and Head is fed by no other block.

- As a consequence of the conditions above, there must be one and only one Head block and one and only one Tail block in a cascade combo.
- A group of physically chained blocks can form more than one cascade combo. For instance, [Figure 31-40](#) shows the same physical chain, but now with blocks 1 and 2 configured as Head and Tail, respectively, forming one combo. The remaining blocks form another combo starting with block 3 (Head), and ending with block N (Tail). Note that the chain-in inputs of blocks from block 3 on are used to carry output data from the first cascaded combo (Tail block 2) to a PSI master block.
- Blocks not used in a cascaded chain can be used normally, isolated (DEFILTER_MCR field CASCD[1:0] = 00), as exemplified in [Figure 31-41](#).
- The optional connection show from block N to block 1 in [Figure 31-39](#) allows block N to be configured as Head or Middle, feeding block 1 configured as Middle or Tail, yielding more flexibility, as in the last example of [Figure 31-41](#).

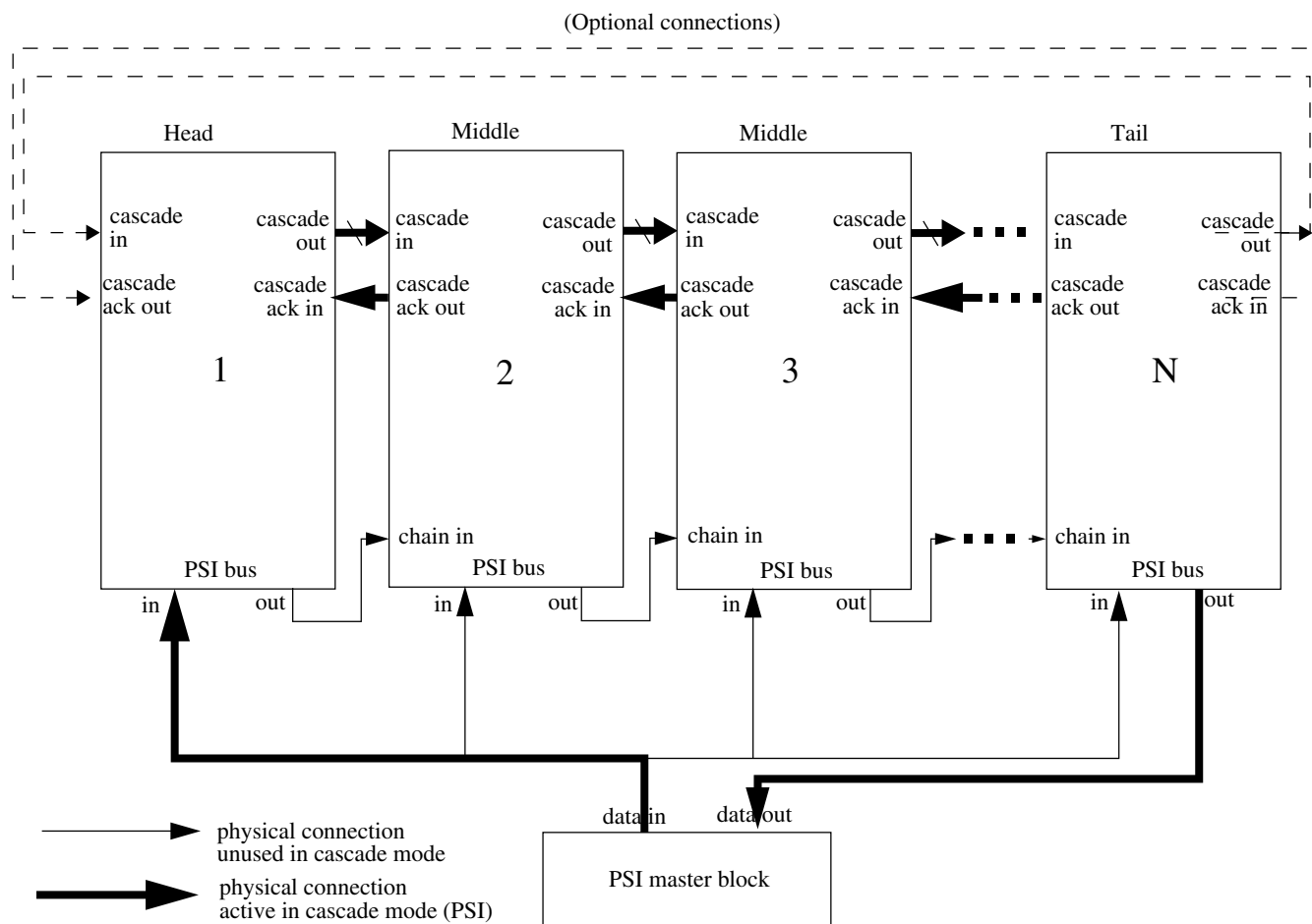


Figure 31-39. Cascade Mode Chain Structure

The input to a cascaded configuration is selected by the DECFILTER_MCR bit ISEL of the Head block. The output target of the cascaded blocks is selected by the DECFILTER_MCR bit ISEL of the Tail block, with the same values used for input selection (ISEL=0 for PSI, ISEL=1 for device slave-bus). DECFILTER_MCR bit MIXM must be written 0 for all cascaded blocks.

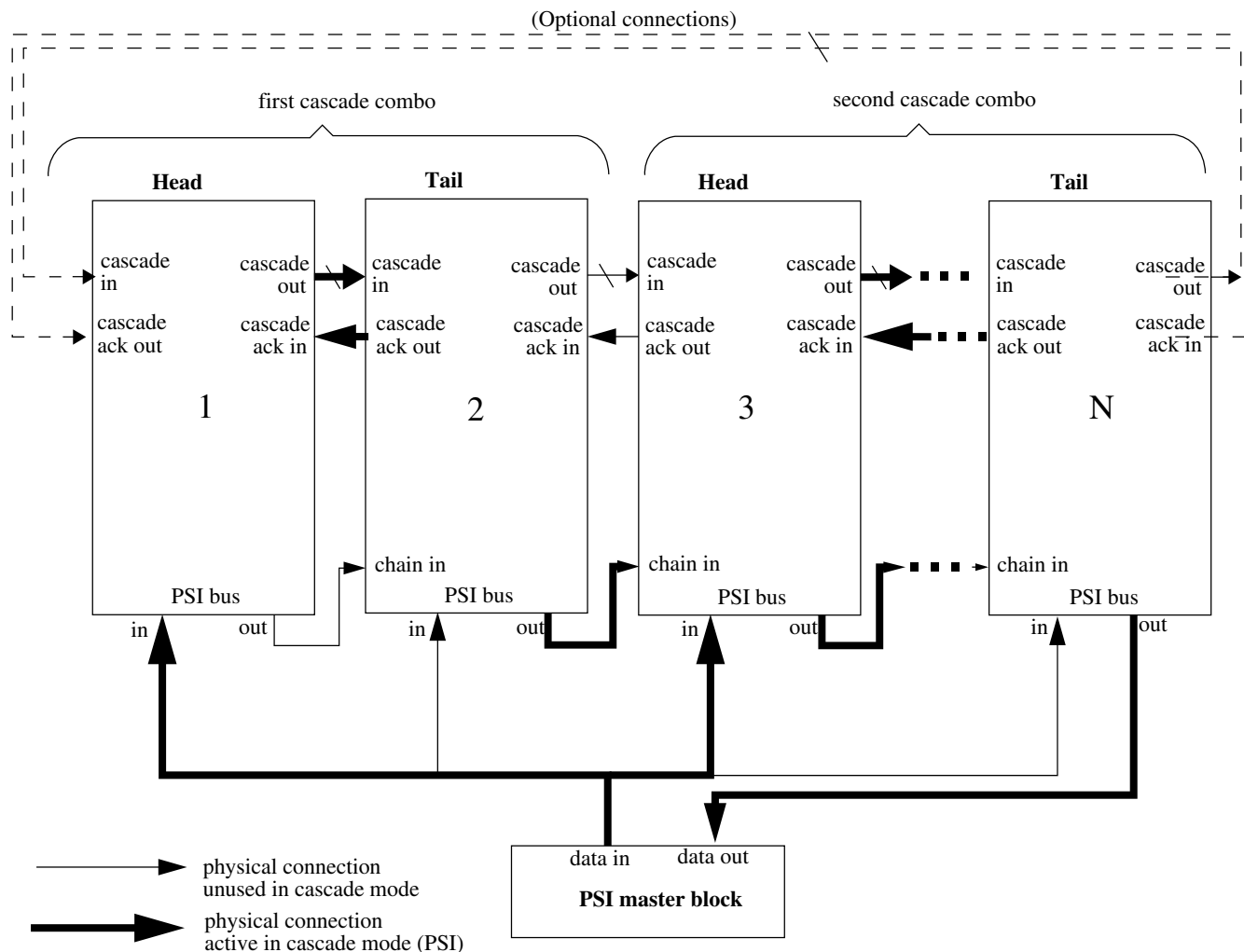
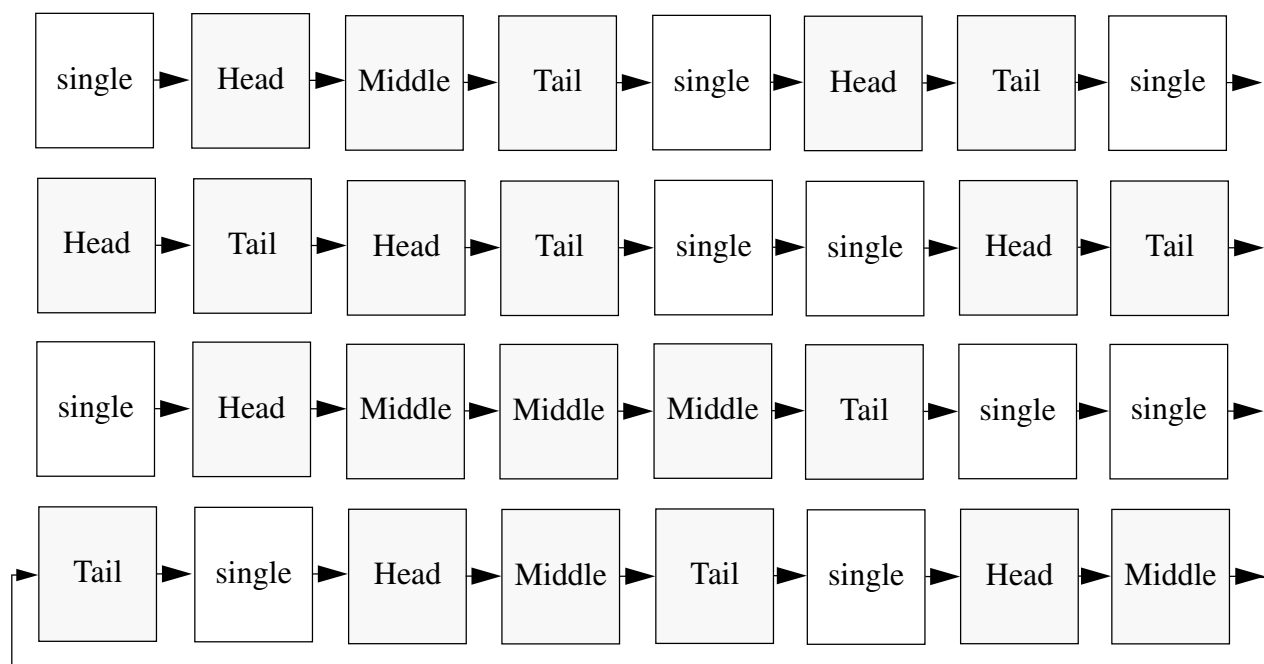


Figure 31-40. Multiple Cascade Mode Chain Structure



(arrows show just the physical cascading order)

Figure 31-41. Examples of mixed Cascaded and single blocks

31.5.16.1 Cascade Freeze, Stop, and Configuration Change Procedures

To change a block configuration mode to or from cascade mode, the following safe procedures must be observed:

- To modify a cascade combo, either to single or any other cascade combo combination, all the cascade combo blocks must have their inputs disabled (using DECFILTER_MCR bit IDIS), in order, from the Head to the Tail block. After a block IDIS bit has been set to 1 (one), one must wait for its DECFILTER_MSR bit BSY to be 0 (zero) before disabling the input of the next block in the sequence.
- Each block in a new cascade combo must be configured with its input disabled. When the mode configuration is done, the combo blocks must have their inputs enabled in order, from the Tail towards the Head block.
- A single block must also be reconfigured the same way, to or from a cascade combo configuration: first disabling its input, and then waiting for a non-busy state before writing DECFILTER_MCR field CASCD.

To take cascade combo blocks to or from freeze or low power modes, a similar procedure must be used:

- Take the Head to freeze or low-power first, wait for DECFILTER_MSR bit BSY=0, and repeat the procedure for the other blocks in the chain in sequence, towards the Tail block.
- Take the blocks out of freeze or low-power modes in the inverse sequence, from Tail to Head.

31.5.17 Return PSI (RPSI) and PSI Output Selector (PSIOSEL)

This block can be connected to two master blocks through two separated Parallel Side Interfaces - PSI0 and PSI1. The input data can come from any of these PSIs and can also come from the deviceslave-bus by DMA or interrupt request. And the filter output result can be sent to any of these PSIs or can be read by the deviceslave-bus by DMA or interrupt request.

The input data can come from PSI0 or PSI1. The PSI data bus structure is presented in [Parallel Side Interface \(PSI\) sub-block description](#)," and it contains the bit field RPSI that defines which PSI output should send the filter result back to its master block. This means the filter result correspondent to some sample data from a PSI input can be sent to any PSI output by the received RPSI value. For example, when the input data comes from PSI1 with RPSI=0 (=1) the filter result is sent back by using PSI1 (PSI0) by asserting the read request signal .

In case of PSI output mixed mode, the input data is supplied by writes to the DEC_FILTER_IB register and the output PSI is directly selected by the bit field PSIOSEL. This is different from RPSI because the sample data does not come from a PSI. For example, when the input data comes with PSIOSEL=0 (=1) the filter result is sent back by using PSI0 (PSI1) by asserting the read request signal .

31.6 Initialization Information

Following are some simple initialization steps to be done before using the decimation filter block. These steps assume that the user has already calculated the filter coefficients using a filter design tool.

31.6.1 Initialization Procedure

The sequence of steps for the block initialization is as follows:

1. Program the configuration registers `DECFILTER_MCR` and `DECFILTER_MXCR` as desired for your application.
2. Write all filter coefficient registers `DECFILTER_COEFn` with the previously calculated values.
3. Enable the filter input, writing `DECFILTER_MCR` bit `IDIS=0`.
4. Run a soft-reset cycle if necessary.
5. The module is ready to receive data from `PSI0/PSI1` or from the device slave-bus interface.

31.7 Application Information

31.7.1 eQADC IP as the PSI Master Block

The system block diagram for the eQADC application is shown in [Figure 31-42](#). In this case, the Decimation Filter receives conversion results generated by one eQADC block using `PSI0` interface. These results can be generated from several different ADC setup configurations which are identified by a specific eQADC Control address within a Conversion command. Conversion commands with Register Address set to zero use the standard configuration setup. The samples generated by the standard configuration setup are sent to one of the local eQADC RFIFO buffers. The samples generated by the Alternate Configurations can be sent to the internal RFIFO or to the eQADC dedicated slave-bus interface (Parallel Side Interface `PSI`) to communicate with the external Decimation Filter IP block or any other block that can communicate with this interface. A bit field in the Alternate Configuration Control Register selects the Internal RFIFO or this slave-bus interface as the destination for the conversion result.

The eQADC can send either conversion data or timestamp data. The conversion data is filtered by the decimation filter and the timestamp is bypassed and sent back to the eQADC.

In the eQADC application, the `TAG` field is used to address the appropriate RFIFO in the eQADC block. In this case, only addresses 0–5 are used since there are only six RFIFOS available in the eQADC block.

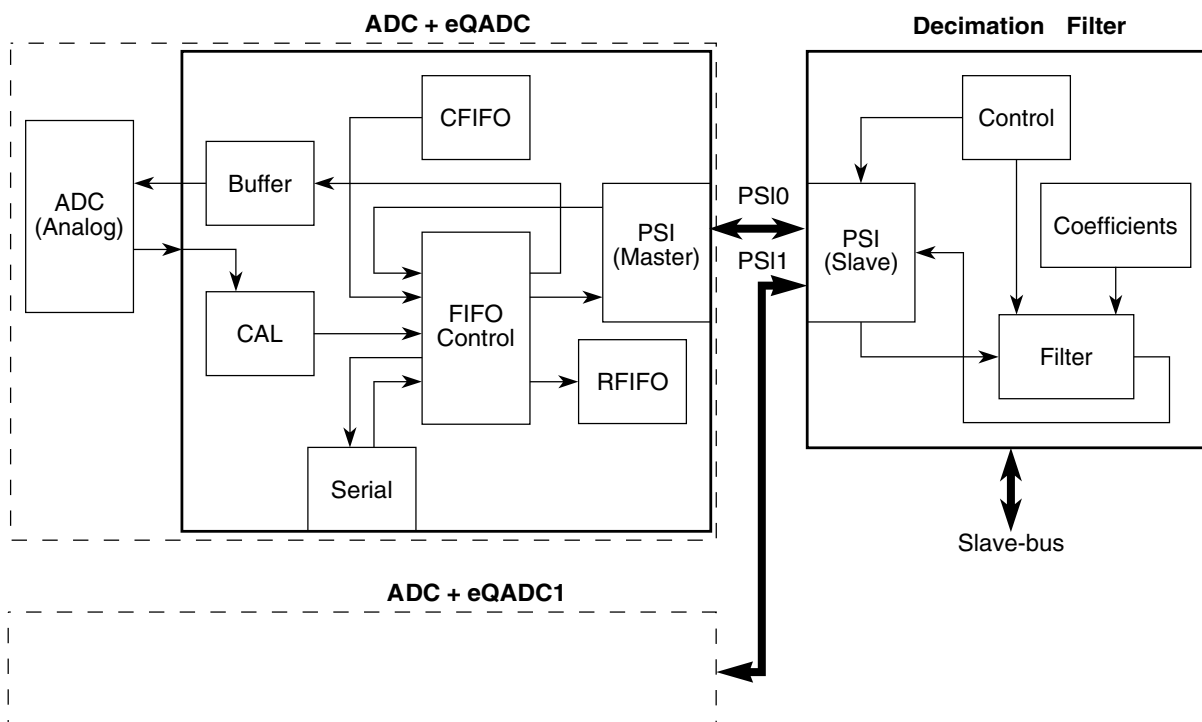


Figure 31-42. Decimation Filter/eQADC interface

31.8 Filter Example Simulation

The decimation filter block operation was checked in a Verilog simulation using calculated filter coefficient values and noisy input data. The expected output values and the RMS error were then calculated.

31.8.1 Coefficients Calculation

The coefficients were calculated using a digital filter design tool. We have supplied some hypothetical filter parameters to the tool and obtained the filter coefficients. The input parameters are:

- Filter characteristics: elliptic/low pass
- Filter type: 4th order IIR
- Input sample rate: 800k sample/s
- Passband edge: 100 kHz

- Stopband edge: 150 kHz
- Passband attenuation: ≤ 1 dB

The software tool gives the IIR filter coefficients in the Z-transform format expressed by Figure 31-43 :

$$\frac{Y(s)}{X(s)} = \frac{B_0 + B_1s + B_2s^2 + B_3s^3 + B_4s^4}{A_0 + A_1s + A_2s^2 + A_3s^3 + A_4s^4}$$

Figure 31-43. Eqn-3

The coefficient results in fixed point decimal representation are shown in Table 31-36.

Table 31-36. Coefficient Values Given by SPW Digital Filter Design Tool

Coefficient	Decimal Value	Coefficient	Decimal Value
B0	0.0221455	A0	1.0
B1	0.00445582948893748	A1	-2.69772868375858
B2	0.0318517846509088	A2	3.234056294853
B3	0.00445582948893748	A3	-1.92028561712454
B4	0.0221455	A4	0.47939080709495

Comparing Figure 31-35 with Figure 31-43, we obtain the relationship between the calculated values of coefficients and the values to be loaded in the DECFILTER_COEFn registers. Note that A0=1 in both cases and that the coefficients A1 to A4 in Figure 31-43 has the opposite signal from Figure 31-35.

See Table 31-37 below to obtain the coefficients. A scale factor of eight is used, being the smallest divider factor to have all coefficient values in the range ($-1 \leq \text{Coef} < +1$).

Table 31-37. Coefficient Values for Decimation Filter

SCALE	S=1	S=8	
COEFn	Decimal Value	Decimal Value	Hexadecimal Values (24 bits)
Coef0 = B0/S	0.0221455	0.00276815891266	0x005AB5
Coef1 = B1/S	0.00445582948893748	0.00055694580078	0x001240
Coef2 = B2/S	0.0318517846509088	0.00398147106171	0x008277
Coef3 = B3/S	0.00445582948893748	0.00055694580078	0x001240
Coef4 = B4/S	0.0221455	0.00276815891266	0x005AB5
Coef5 = -A1/S	2.69772868375858	0.33721613883972	0x2B29E6
Coef6 = -A2/S	-3.234056294853	-0.40425717830658	0xCC414E
Coef7 = -A3/S	1.92028561712454	0.24003565311432	0x1EB97D
Coef8 = -A4/S	-0.47939080709495	-0.05992400646210	0xF8546A

31.8.2 Input Data Calculation

The 24-bit words of input data are samples of a sum of two tones: one tone of 30kHz and another tone of 250kHz. The samples were calculated at the rate of 800k samples per second. The tones have the same amplitude and it is assured that the resultant amplitude is smaller than 1 so as to obtain samples in the range $(-1 \leq \text{sample} < +1)$. It is supposed the input data are signed values in the two's complement format in the range $(-1 \leq \text{sample} < +1)$.

31.8.3 Filter Results

The decimation filter was simulated using the calculated coefficients and the input data samples. A scaling factor of eight in the configuration register `DECFILTER_MCR`, and no decimation factor, were used to obtain the maximum of output results from the filter.

The theoretical expected values from this filter were also calculated, and these results were compared with those from the decimation filter. The resultant RMS error when considering about 500 samples was about -97 dB.

Chapter 32

Temperature Sensor (TSENS)

32.1 Chip-specific Temperature Sensor information

This chip has two instances of the Temperature Sensor.

32.1.1 Temperature Sensor programming control

The control and status registers for the Temperature Sensor are implemented in the PMC digital interface.

32.1.2 Accessing TSC parameters

The Temperature Sensor Calibration (TSC) parameters, including TSCA and TSCB, are used to calculate device temperature.

See the UTEST flash memory map for the specific locations of the TSC parameter values.

32.1.3 Temperature name alignment

The Temperature Sensor modules and Power Management Controller (PMC) module use different terminology to refer to the same temperature points. The following table aligns the terminology.

Table 32-1. Temperature Sensor and PMC temperature terminology

Temperature	Temperature Sensor name	PMC name
-40°C	Low temperature	Temperature 0
150°C	High temperature	Temperature 2

32.2 Introduction

The device includes two on-chip temperature sensors. The temperature sensor module monitors device temperature and delivers one analog output signal and two digital output signals each (under- and over-temperature flags). The analog output consists of a voltage signal directly proportional to the internal junction temperature. The analog output is connected to an input channel of an ADC on the device (see the ADC details for the channel number). The internal junction temperature must be calculated by software based on the sampled temperature sensor voltage; sampled bandgap reference voltage, which comes from another module; and calibration parameter values stored in internal flash memory. The two digital outputs, connected to the PMC module, are used to signal under- and over-temperature operating conditions.

The digital outputs signal when the junction temperature drifts below the low temperature threshold or above the high temperature threshold. These signals notify the device to take action to appropriately adjust the device temperature in response to an out-of-specification low or high temperature operating condition. Calibration parameter values associated with the temperature threshold detection feature are determined and stored in internal flash memory during production testing at the factory.

32.3 Functional Description

The temperature sensor generates one analog output voltage, $V_{\text{TSENS}}(T)$, which is proportional to the absolute current junction temperature of the device, and two digital outputs that signal whether the junction temperature has reached either a pre-set low temperature threshold or a pre-set high temperature threshold.

An on-chip ADC module is used to convert the analog output voltage, as well as the bandgap reference voltage, into a digital representation. These values, along with parameter values stored in on-chip flash memory, are used by software to calculate the device junction temperature.

32.3.1 Temperature threshold detection (digital output generation)

Temperature threshold trimming values are adjusted through calibration during factory test and stored in on-chip flash memory. While enabled, if a new trimming value is loaded into the Temperature Sensor module, it may take up to 1 microsecond for the digital outputs to be updated to reflect the new trimming condition.

When the temperature threshold detection feature is enabled, the temperature sensor monitors the internal junction temperature of the chip and asserts a signal if any of the following temperature thresholds are crossed.

- The low temperature digital output signals if the junction temperature falls below the low temperature threshold (-40°C).
- The high temperature digital output signals if the junction temperature rises above the high temperature threshold (150°C).

The threshold detection circuit is calibrated with respect to the factory-test temperatures: the low temperature threshold is trimmed to match the cold insertion test temperature (nominally equal to -40°C), and the high temperature threshold is trimmed to match the hot insertion test temperature (nominally equal to 150°C). In other words, the thresholds are not trimmed to absolute values of temperature but to the specific temperatures for which the device is validated during factory test.

Hysteresis is applied whenever outputs transition to a logic level low in order to eliminate spurious toggling. Digital output behavior is illustrated on [Figure 32-1](#).

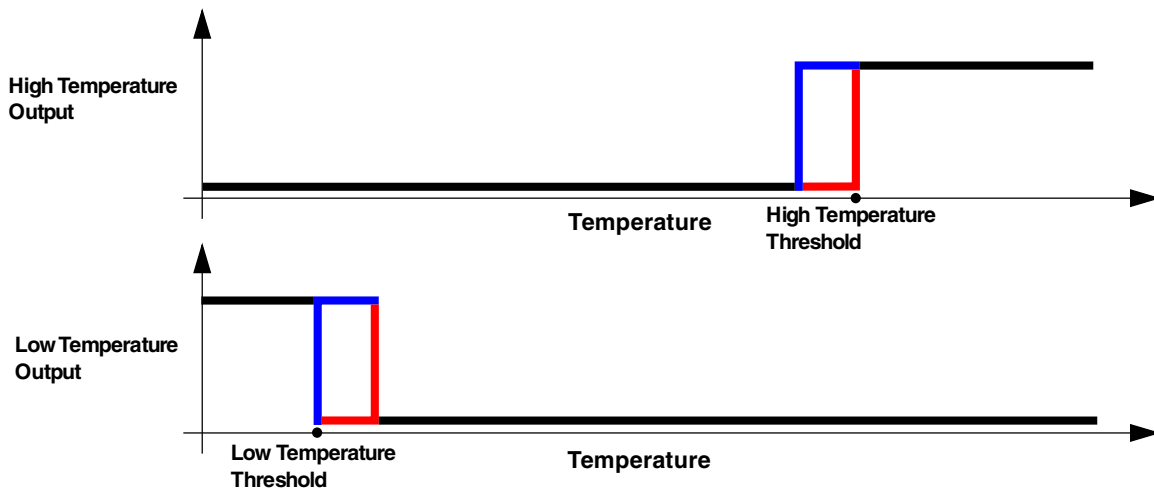


Figure 32-1. Digital output behavior with temperature

32.3.2 Linear temperature sensor (analog output generation)

When analog output generation is enabled, the temperature sensor outputs a voltage proportional to the internal junction temperature of the chip. This analog voltage signal is converted into a digital code by an on-chip ADC. The temperature value is obtained from a linear voltage-temperature relation with coefficients adjusted by calibration parameters extracted during factory test and programmed into flash memory.

32.4 Temperature formula

The system chain that translates device junction temperature into a digital variable is composed of the temperature sensor, a bandgap reference voltage source and the on-chip ADC. Both analog output voltages of the temperature sensor and the bandgap reference voltage source must be converted by the ADC into digital codes to obtain the device junction temperature. The temperature formula shown in [Figure 32-2](#) is used to calculate internal device junction temperature.

$$T = (232 + TSCA \times 2^{-6}) \times \frac{V_{TSENS\_CODE}(T)}{V_{BG\_CODE}(T)} - (273 + TSCB \times 2^{-4}) [^{\circ}\text{C}]$$

Where:

$$TSCA = \left(\frac{T_{HIGH} - T_{LOW}}{V_{TSENS\_CODE}(T_{HIGH}) - V_{TSENS\_CODE}(T_{LOW})} \times V_{BG\_CODE}(T_{FINAL}) - 232 \right) \times 2^6$$

$$TSCB = \left(\frac{T_{HIGH} - T_{LOW}}{V_{TSENS\_CODE}(T_{HIGH}) - V_{TSENS\_CODE}(T_{LOW})} \times V_{TSENS\_CODE}(T_{LOW}) - T_{LOW} - 273 \right) \times 2^4$$

$$V_{TSENS\_CODE}(T_{LOW}) = \frac{V_{TSENS}(T_{LOW})}{V_{ref0}} \times 2^{12}$$

$$V_{TSENS\_CODE}(T_{HIGH}) = \frac{V_{TSENS}(T_{HIGH})}{V_{ref0}} \times 2^{12}$$

$$V_{BG\_CODE}(T_{FINAL}) = \frac{V_{BG}(T_{FINAL})}{V_{ref0}} \times 2^{12}$$

$$V_{BG\_CODE}(T) = \frac{V_{BG}(T)}{V_{ref}} \times 2^{12}$$

$$V_{TSENS\_CODE}(T) = \frac{V_{TSENS}(T)}{V_{ref}} \times 2^{12}$$

Notes:

- TSCA and TSCB are 16-bit signed integer constants stored in flash memory during factory test and calibration
- $V_{TSENS}(T)$ is the temperature sensor output sampled by the ADC
- $V_{BG}(T)$ is the bandgap reference voltage sampled by the ADC at temperature T
- V_{ref} is the ADC reference voltage in normal usage of the device
- V_{ref0} is the ADC reference voltage during factory calibration
- T_{LOW} is the low temperature factory calibration temperature
- T_{HIGH} is the hot temperature factory calibration temperature
- T_{FINAL} is the final insertion factory calibration temperature

Figure 32-2. Temperature formula

Parameters TSCA and TSCB are 16-bit signed integer constants stored during factory test and calibration that need to be used in the temperature formula to calculate device junction temperature. The values for TSCA and TSCB translate into positive or negative shifts of the linear relation coefficients that compensate for manufacturing process variation. [Figure 32-2](#) shows how parameters TSCA and TSCB are calculated in terms of the actual variables measured during factory calibration.

32.5 Calculating device temperature

Software must perform the following steps in order to calculate device temperature.

Calculating device temperature

1. Measure the bandgap reference voltage using the on-chip ADC module and calculate $V_{BG\_CODE}(T)$ according to the formula shown in [Figure 32-2](#).
2. Measure the temperature sensor output voltage using the on-chip ADC module and calculate $V_{TSENS\_CODE}(T)$ according to the formula shown in [Figure 32-2](#).
3. Retrieve parameters TSCA and TSCB from flash memory.
4. Calculate T according to the formula shown in [Figure 32-2](#).

Note

Temperature Formula may be calculated using signed 32-bit (integer) operations showing negligible precision loss.

Chapter 33

System Timer Module (STM)

33.1 Chip-specific STM information

This chip has one instance of the STM.

33.2 Introduction

This section provides an overview, list of features, and modes of operation for the STM.

33.2.1 Overview

The System Timer Module (STM) is a 32-bit timer designed to support commonly required system and application software timing functions. The STM includes a 32-bit up counter and four 32-bit compare channels with a separate interrupt source for each channel. The counter is driven by the system clock divided by an 8-bit prescale value (1 to 256).

33.2.2 Features

The STM has the following features:

- One 32-bit up counter with 8-bit prescaler
- Four 32-bit compare channels
- Independent interrupt source for each channel
- Counter can be stopped in debug mode

33.2.3 Modes of operation

The STM supports two device modes of operation: normal and debug. When the STM is enabled in normal mode, its counter runs continuously. In debug mode, operation of the counter is controlled by the FRZ bit in the STM_CR register. If the FRZ bit is set, the counter is stopped in debug mode, otherwise it continues to run.

33.3 External signal description

The STM does not have any external interface signals.

33.4 Memory map and registers

The STM programming model includes a group of 32-bit registers—a module control register, a counter value register, and three registers for each channel.

The STM registers can only be accessed using 32-bit (word) accesses. Attempted references using a different size or to a reserved address generates a bus error termination.

STM memory map

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/page
0	STM Control Register (STM_CR)	32	R/W	0000_0000h	33.4.1/1347
4	STM Count Register (STM_CNT)	32	R/W	0000_0000h	33.4.2/1348
10	STM Channel Control Register (STM_CCR0)	32	R/W	0000_0000h	33.4.3/1348
14	STM Channel Interrupt Register (STM_CIR0)	32	w1c	0000_0000h	33.4.4/1349
18	STM Channel Compare Register (STM_CMP0)	32	R/W	0000_0000h	33.4.5/1349
20	STM Channel Control Register (STM_CCR1)	32	R/W	0000_0000h	33.4.3/1348
24	STM Channel Interrupt Register (STM_CIR1)	32	w1c	0000_0000h	33.4.4/1349
28	STM Channel Compare Register (STM_CMP1)	32	R/W	0000_0000h	33.4.5/1349
30	STM Channel Control Register (STM_CCR2)	32	R/W	0000_0000h	33.4.3/1348
34	STM Channel Interrupt Register (STM_CIR2)	32	w1c	0000_0000h	33.4.4/1349
38	STM Channel Compare Register (STM_CMP2)	32	R/W	0000_0000h	33.4.5/1349
40	STM Channel Control Register (STM_CCR3)	32	R/W	0000_0000h	33.4.3/1348
44	STM Channel Interrupt Register (STM_CIR3)	32	w1c	0000_0000h	33.4.4/1349
48	STM Channel Compare Register (STM_CMP3)	32	R/W	0000_0000h	33.4.5/1349

33.4.1 STM Control Register (STM_CR)

The STM Control Register includes the prescale value, freeze control and timer enable bits.

Address: 0h base + 0h offset = 0h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	CPS								0							
W															FRZ	TEN
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

STM_CR field descriptions

Field	Description
0–15 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
16–23 CPS	Counter Prescaler. Selects the clock divide value for the prescaler (1 - 256). 0x00 Divide system clock by 1 0x01 Divide system clock by 2 ----- 0xFF Divide system clock by 256
24–29 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
30 FRZ	Freeze. Allows the timer counter to be stopped when the device enters debug mode. NOTE: When the MCU enters debug mode, the STM is notified and uses the FRZ bit to determine counter mode. 0 STM counter continues to run in debug mode. 1 STM counter is stopped in debug mode.
31 TEN	Timer counter Enabled. 0 Counter is disabled. 1 Counter is enabled.

33.4.2 STM Count Register (STM_CNT)

The STM Count Register holds the timer count value.

Address: 0h base + 4h offset = 4h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	CNT																															
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

STM_CNT field descriptions

Field	Description
0–31 CNT	Timer count value used as the time base for all channels. When enabled, the counter increments at the rate of the system clock divided by the prescale value.

33.4.3 STM Channel Control Register (STM_CCRn)

The STM Channel Control Register (STM_CCRn) has the enable bit for channel n of the timer.

Address: 0h base + 10h offset + (16d × i), where i=0d to 3d

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	
R	0																CEN
W																	
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	

STM_CCRn field descriptions

Field	Description
0–30 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
31 CEN	Channel Enable 0 The channel is disabled. 1 The channel is enabled.

33.4.4 STM Channel Interrupt Register (STM_CIR_n)

The STM Channel Interrupt Register has the interrupt flag for channel *n* of the timer.

Address: 0h base + 14h offset + (16d × *i*), where *i*=0d to 3d

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0															CIF
W																w1c
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

STM_CIR_n field descriptions

Field	Description
0–30 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
31 CIF	Channel Interrupt Flag 0 No interrupt request. 1 Interrupt request due to a match on the channel.

33.4.5 STM Channel Compare Register (STM_CMP_n)

The STM channel compare register (STM_CMP_n) holds the compare value for channel *n*.

Address: 0h base + 18h offset + (16d × *i*), where *i*=0d to 3d

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	CMP																															
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	

STM_CMP_n field descriptions

Field	Description
0–31 CMP	Compare value for channel <i>n</i> . If the STM_CCR _n [CEN] bit is set and the STM_CMP _n register matches the STM_CNT register, a channel interrupt request is generated and the STM_CIR _n [CIF] bit is set.

STM_CMPn field descriptions (continued)

Field	Description

33.5 Functional description

The STM has one 32-bit up counter (STM_CNT) that is used as the time base for all channels. When enabled, the counter increments at the system clock frequency divided by a prescale value. The STM_CR[CPS] field sets the divider to any value in the range from 1 to 256. The counter is enabled with the STM_CR[TEN] bit. When enabled in normal mode, the counter continuously increments. When enabled in debug mode, the counter operation is controlled by the STM_CR[FRZ] bit. When the STM_CR[FRZ] bit is set, the counter is stopped in debug mode; otherwise, it continues to run in debug mode. The counter rolls over at 0xFFFF_FFFF to 0x0000_0000 with no restrictions at this boundary.

The STM has four identical compare channels. Each channel includes a channel control register (STM_CCRn), a channel interrupt register (STM_CIRn), and a channel compare register (STM_CMPn). The channel is enabled by setting the STM_CCRn[CEN] bit. When enabled, the channel will set the STM_CIRn[CIF] bit and generate an interrupt request when the channel compare register matches the timer counter. The interrupt request is cleared by writing 1 to the STM_CIRn[CIF] bit. A write of 0 to the STM_CIRn[CIF] bit has no effect.

NOTE

The STM counter does not advance when the system clock is stopped.

Chapter 34

Software Watchdog Timer (SWT)

34.1 Chip-specific SWT information

This chip has two instances of the SWT module: SWT_A and SWT_B.

34.1.1 SWT register reset values

The following table identifies chip-specific reset values of SWT registers.

Table 34-1. Chip-specific SWT register reset values

Register	SWT_A	SWT_B
CR	FF00_010Bh ¹	FF00_010Ah
TO	0005_FCD0h	0005_FCD0h

1. This value is present after a hardware reset and before the BAM executes. The BAM code (RCHW) might change it before application code executes. See [Reset Configuration Half Word \(RCHW\)](#) for more information.

34.1.2 SWT time-out events and invalid accesses: configuring reset and interrupt requests

To configure the results of SWT time-out events and invalid accesses for an SWT that is enabled (SWT_CR[WEN] is 1):

- Program these [SWT Control Register \(SWT_CR\)](#) fields:

- **ITR** determines when the outcome of a time-out event is a reset request or an interrupt request.
- **RIA** determines when the outcome of an invalid access is a reset request or a bus error.
- Program FCCU reactions to the non-critical faults (NCFs) that are associated with these reset and interrupt requests.

See **FCCU fault inputs** for the default reaction to each NCF.

CAUTION

For any NCF that receives an SWT reset request: If the default FCCU reaction is no reset but the intended application requires a reset, configure the FCCU's reaction to be a reset.

The following table shows the NCFs to which the SWT generates reset and interrupt requests for time-out events.

Table 34-2. Routing of SWT reset and interrupt requests for time-out events

SWT instance	SWT_CR[ITR] is 0	SWT_CR[ITR] is 1	
	Any time-out generates reset request to	Initial time-out generates interrupt request to	Second consecutive time-out generates reset request to
SWT_A	NCF[14]	IRQ[8] and NCF[16]	NCF[14]
SWT_B	NCF[15]	IRQ[501]	NCF[15]

The following table shows the NCFs to which the SWT can generate reset requests for invalid accesses.

Table 34-3. Routing of SWT reset requests for invalid accesses

SWT instance	SWT_CR[RIA] is 0	SWT_CR[RIA] is 1
SWT_A	Invalid access generates bus error	Invalid access generates reset request to NCF[14]
SWT_B	Invalid access generates bus error	Invalid access generates reset request to NCF[15]

34.2 Introduction

This section provides an overview, list of features, and modes of operation for the SWT.

34.2.1 Overview

The Software Watchdog Timer (SWT) is a peripheral module that can prevent system lockup in situations such as software getting trapped in a loop or if a bus transaction fails to terminate. When enabled, the SWT requires periodic execution of a watchdog servicing operation. The servicing operation resets the timer to a specified time-out period. If this servicing action does not occur before the timer expires, the SWT generates an interrupt or hardware reset request. The SWT can be configured to generate a reset request or interrupt on an initial time-out. The SWT always generates a reset request on a second consecutive time-out.

34.2.2 Features

The SWT has the following features:

- 32-bit time-out register to set the time-out period
- Programmable selection of system or oscillator clock, for timer operation
- Programmable selection of window mode or regular servicing
- Programmable selection of reset or interrupt on an initial time-out
- Programmable selection of the servicing mode
- Master access protection
- Hard and soft configuration lock bits
- Reset configuration inputs allow timer to be enabled out of reset

34.2.3 Modes of operation

The SWT supports three device modes of operation: normal, debug and stop. When the SWT is enabled in normal mode, its counter runs continuously. In debug mode, operation of the counter is controlled by the FRZ bit in the SWT_CR. If the FRZ bit is set, the counter is stopped in debug mode, otherwise it continues to run. In stop mode, operation of the counter is controlled by the STP bit in the SWT_CR. If the STP bit is set, the counter is stopped in stop mode, otherwise it continues to run.

34.3 External signal description

The SWT module does not have any external interface signals.

34.4 Memory Map and Registers

The SWT programming model has seven 32-bit registers. The programming model can be accessed using only 32-bit (word) accesses.

Any of the following attempted accesses are invalid:

- References using non-32-bit access sizes
- Writes to read-only registers
- Writes of incorrect values to SWT_SR when the SWT is enabled
- Accesses to reserved addresses
- Accesses by masters without permission

For an enabled SWT, an invalid access generates a reset request if SWT_CR[RIA] is 1. An invalid access generates a bus error if SWT_CR[RIA] is 0.

If either HLK or SLK in SWT_CR is 1, then the SWT_CR, SWT_TO, SWT_WN, and SWT_SK registers are read-only.

SWT memory map

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/page
0	SWT Control Register (SWT_CR)	32	R/W	See section	34.4.1/1355
4	SWT Interrupt Register (SWT_IR)	32	R/W	0000_0000h	34.4.2/1358
8	SWT Time-out Register (SWT_TO)	32	R/W	See section	34.4.3/1359
C	SWT Window Register (SWT_WN)	32	R/W	0000_0000h	34.4.4/1359
10	SWT Service Register (SWT_SR)	32	W	0000_0000h	34.4.5/1360
14	SWT Counter Output Register (SWT_CO)	32	R	0000_0000h	34.4.6/1360
18	SWT Service Key Register (SWT_SK)	32	R/W	0000_0000h	34.4.7/1361

34.4.1 SWT Control Register (SWT_CR)

NOTE

The reset value of this register is implementation specific. See the chip-specific SWT information.

The SWT_CR contains fields for configuring and controlling the SWT.

This register is read-only if either SWT_CR[HLK] or SWT_CR[SLK] is 1.

Address: 0h base + 0h offset = 0h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	MAP0	MAP1	MAP2	MAP3	MAP4	MAP5	MAP6	MAP7	0							
W																
Reset	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*
Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0					SMD		RIA	WND	ITR	HLK	SLK	CSL	STP	FRZ	WEN
W																
Reset	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*

* Notes:

- The reset value of this register is implementation specific. See the chip-specific SWT information.

SWT_CR field descriptions

Field	Description
0 MAP0	Master Access Protection for Master 0. The platform bus master assignments are device specific. The number of this field corresponds to the logical ID of the bus master. MAP0 corresponds to logical IDs 0 and 8. For the masters with the logical IDs to which the MAPn fields correspond, see the chip-specific information for the Crossbar Switch. 0 Access for the master is not enabled 1 Access for the master is enabled
1 MAP1	Master Access Protection for Master 1. The platform bus master assignments are device specific. The number of this field corresponds to the logical ID of the bus master. MAP1 corresponds to logical IDs 1 and 9. For the masters with the logical IDs to which the MAPn fields correspond, see the chip-specific information for the Crossbar Switch. 0 Access for the master is not enabled 1 Access for the master is enabled

Table continues on the next page...

SWT_CR field descriptions (continued)

Field	Description
2 MAP2	Master Access Protection for Master 2. The platform bus master assignments are device specific. The number of this field corresponds to the logical ID of the bus master. MAP2 corresponds to logical IDs 2 and 10. For the masters with the logical IDs to which the MAPn fields correspond, see the chip-specific information for the Crossbar Switch. 0 Access for the master is not enabled 1 Access for the master is enabled
3 MAP3	Master Access Protection for Master 3. The platform bus master assignments are device specific. The number of this field corresponds to the logical ID of the bus master. MAP3 corresponds to logical IDs 3 and 11. For the masters with the logical IDs to which the MAPn fields correspond, see the chip-specific information for the Crossbar Switch. 0 Access for the master is not enabled 1 Access for the master is enabled
4 MAP4	Master Access Protection for Master 4. The platform bus master assignments are device specific. The number of this field corresponds to the logical ID of the bus master. MAP4 corresponds to logical IDs 4 and 12. For the masters with the logical IDs to which the MAPn fields correspond, see the chip-specific information for the Crossbar Switch. 0 Access for the master is not enabled 1 Access for the master is enabled
5 MAP5	Master Access Protection for Master 5. The platform bus master assignments are device specific. The number of this field corresponds to the logical ID of the bus master. MAP5 corresponds to logical IDs 5 and 13. For the masters with the logical IDs to which the MAPn fields correspond, see the chip-specific information for the Crossbar Switch. 0 Access for the master is not enabled 1 Access for the master is enabled
6 MAP6	Master Access Protection for Master 6. The platform bus master assignments are device specific. The number of this field corresponds to the logical ID of the bus master. MAP6 corresponds to logical IDs 6 and 14. For the masters with the logical IDs to which the MAPn fields correspond, see the chip-specific information for the Crossbar Switch. 0 Access for the master is not enabled 1 Access for the master is enabled
7 MAP7	Master Access Protection for Master 7. The platform bus master assignments are device specific. The number of this field corresponds to the logical ID of the bus master. MAP7 corresponds to logical IDs 7 and 15. For the masters with the logical IDs to which the MAPn fields correspond, see the chip-specific information for the Crossbar Switch.

Table continues on the next page...

SWT_CR field descriptions (continued)

Field	Description
	0 Access for the master is not enabled 1 Access for the master is enabled
8–20 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
21–22 SMD	Service Mode 00 Fixed Service Sequence: The watchdog is serviced by writing the fixed sequence 0xA602, 0xB480 to SWT_SR. 01 Keyed Service Sequence: The watchdog is serviced by writing two pseudorandom key values to SWT_SR. 10 Reserved—do not use. Writing a value can cause the SWT not to be serviced. 11 Reserved—do not use. Writing this value can cause the SWT not to be serviced.
23 RIA	Reset on Invalid Access NOTE: For a description of how this chip implements SWT reset requests resulting from SWT invalid accesses, see the chip-specific SWT information. 0 Invalid access to the SWT generates a bus error 1 Invalid access to the SWT causes a reset request if WEN is 1
24 WND	Window Mode 0 Regular mode: service sequence can be done at any time 1 Windowed mode: service sequence is valid only when the down counter is less than the value in the SWT_WN register
25 ITR	Interrupt Then Reset NOTE: For a description of how this chip implements SWT reset requests and interrupt requests resulting from SWT time-outs, see the chip-specific SWT information. 0 Generate a reset request on a time-out 1 Generate an interrupt on an initial time-out; generate a reset request on a second consecutive time-out
26 HLK	Hard Lock This bit is cleared only at reset. 0 SWT_CR, SWT_TO, SWT_WN and SWT_SK are read/write registers if SLK is 0 1 SWT_CR, SWT_TO, SWT_WN and SWT_SK are read-only registers
27 SLK	Soft Lock This bit is cleared by writing the unlock sequence to the service register. 0 SWT_CR, SWT_TO, SWT_WN and SWT_SK are read/write registers if HLK is 0 1 SWT_CR, SWT_TO, SWT_WN and SWT_SK are read-only registers
28 CSL	Clock Selection Selects the clock that drives the internal timer. 0 System clock 1 Oscillator clock

Table continues on the next page...

SWT_CR field descriptions (continued)

Field	Description
29 STP	Stop Mode Control Allows the watchdog timer to be stopped when the device enters stop mode. 0 SWT counter continues to run in stop mode 1 SWT counter is stopped in stop mode
30 FRZ	Debug Mode Control Allows the watchdog timer to be stopped when the device enters debug mode. 0 SWT counter continues to run in debug mode 1 SWT counter is stopped in debug mode
31 WEN	Watchdog Enabled 0 SWT is disabled 1 SWT is enabled

34.4.2 SWT Interrupt Register (SWT_IR)

The SWT_IR contains the time-out interrupt flag.

Address: 0h base + 4h offset = 4h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15													
R	0																												
W																													
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0													
Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31													
R	0																												TIF
W																													w1c
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0													

SWT_IR field descriptions

Field	Description
0–30 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
31 TIF	Time-out Interrupt Flag The flag and interrupt are cleared by writing a 1 to this bit. Writing a 0 has no effect. 0 No interrupt request 1 Interrupt request due to an initial time-out

34.4.3 SWT Time-out Register (SWT_TO)

NOTE

The reset value of this register is implementation specific. See the chip-specific SWT information.

The SWT Time-out (SWT_TO) register contains the 32-bit time-out period. This register is read-only if either SWT_CR[HLK] or SWT_CR[SLK] is 1.

Address: 0h base + 8h offset = 8h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	WTO																															
W																																
Reset	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	

* Notes:

- The reset value of this register is implementation specific. See the chip-specific SWT information.

SWT_TO field descriptions

Field	Description
0–31 WTO	Watchdog time-out period in clock cycles. An internal 32-bit down counter is loaded with this value or 0x100, whichever is greater, when the service sequence is written or when the SWT is enabled.

34.4.4 SWT Window Register (SWT_WN)

The SWT Window (SWT_WN) register contains the 32-bit window start value. This register is cleared on reset. This register is read-only if either SWT_CR[HLK] or SWT_CR[SLK] is 1.

Address: 0h base + Ch offset = Ch

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	WST																															
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	

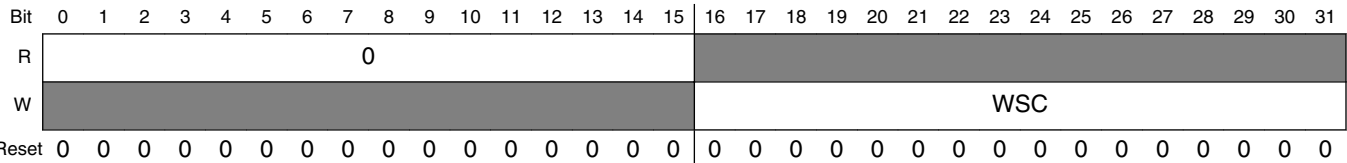
SWT_WN field descriptions

Field	Description
0–31 WST	Window Start Value When window mode is enabled, the service sequence can only be written when the internal down counter is less than this value.

34.4.5 SWT Service Register (SWT_SR)

The SWT Service (SWT_SR) register is the target for service operation writes used to reset the watchdog timer.

Address: 0h base + 10h offset = 10h



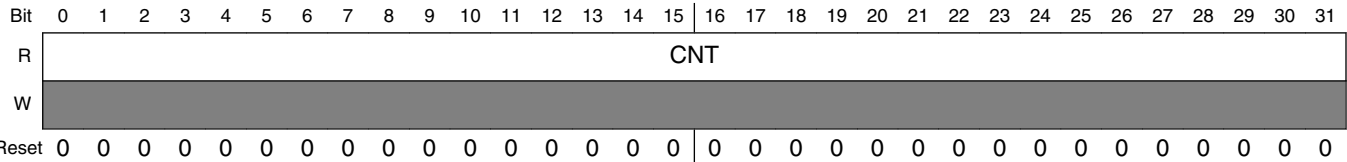
SWT_SR field descriptions

Field	Description
0–15 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
16–31 WSC	Watchdog Service Code This field is used to service the watchdog and to clear the Soft Lock bit (SWT_CR[SLK]). If the SWT_CR[SMD] field is 01b, two pseudorandom key values are written to service the watchdog; see Functional description for details. Otherwise, the sequence 0xA602 followed by 0xB480 is written to the WSC field. To clear the Soft Lock bit (SWT_CR[SLK]), the value 0xC520 followed by 0xD928 is written to the WSC field. When read, the WSC field always returns zero.

34.4.6 SWT Counter Output Register (SWT_CO)

The SWT Counter Output (SWT_CO) register is a read-only register that shows the value of the internal down counter when the SWT is disabled.

Address: 0h base + 14h offset = 14h



SWT_CO field descriptions

Field	Description
0–31 CNT	Watchdog Count When the watchdog is disabled (SWT_CR[WEN] is 0), this field shows the value of the internal down counter. When the watchdog is enabled (SWT_CR[WEN] is 1), this field is cleared (the value is

SWT_CO field descriptions (continued)

Field	Description
	0x0000_0000). Values in this field can lag behind the internal counter value for up to 6 system clock cycles plus 8 counter clock cycles. Therefore, the value read from this field immediately after disabling the watchdog may be higher than the actual value of the internal counter.

34.4.7 SWT Service Key Register (SWT_SK)

The SWT Service Key (SWT_SK) register holds the previous (or initial) service key value. This register is read-only if either SWT_CR[HLK] or SWT_CR[SLK] is 1.

Address: 0h base + 18h offset = 18h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0																SK															
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	

SWT_SK field descriptions

Field	Description
0–15 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
16–31 SK	Service Key This field is the previous (or initial) service key value used in keyed service mode. If SWT_CR[SMD] is 01b, the next key value to be written to the SWT_SR is $(17 \cdot SK + 3) \bmod 2^{16}$.

34.5 Functional description**34.5.1 Introduction**

The SWT is a 32-bit window watchdog timer designed to enable the system to recover in situations such as software getting trapped in a loop or if a bus transaction fails to terminate. It includes:

- a control register (SWT_CR),
- an interrupt register (SWT_IR),
- a time-out register (SWT_TO),
- a window register (SWT_WN),
- a service register (SWT_SR),

- a counter output register (SWT_CO), and
- a service key register (SWT_SK).

Accesses to SWT registers occur with no peripheral bus wait states. (The peripheral bus bridge may add one or more system wait states.) However, due to synchronization logic in the SWT design, recognition of the service sequence or configuration changes may require up to three system plus seven counter clock cycles.

34.5.1.1 SWT_CR register

The SWT_CR register includes bits to enable the timer, set configuration options, and lock configuration of the module. The watchdog is enabled by setting the SWT_CR[WEN] bit. The reset value of the SWT_CR[WEN] bit is device specific.³ If the reset value of this bit is 1, the watchdog starts operation automatically after reset is released. Some devices can be configured to clear this bit automatically during the boot process.

34.5.1.2 SWT_TO register

The SWT_TO register holds the watchdog time-out period in clock cycles unless the value is less than 0x100, in which case the time-out period is set to 0x100. When the SWT is enabled, the time-out period is loaded into an internal 32-bit down counter each time a valid service operation is performed. The SWT_CR[CSL] bit selects which clock (system or oscillator) is used to drive the down counter.

34.5.1.3 SWT_CO register

The SWT_CO register shows the value of the down counter when the watchdog is disabled. When the watchdog is enabled this register is cleared. The value shown in this register can lag behind the value in the internal counter for up to six system plus eight counter clock cycles.

The SWT_CO register can be used during a software self test of the SWT. For example, the SWT can be enabled and not serviced for a fixed period of time less than the time-out value. Then the SWT can be disabled (SWT_CR[WEN] cleared) and the value of the SWT_CO register read to determine if the internal down counter is working properly.

3. See the chip-specific SWT information.

34.5.2 Configuration locking

The configuration of the SWT can be locked through use of either a soft lock or a hard lock. In either case, when locked, the SWT_CR, SWT_TO, SWT_WN and SWT_SK registers are read-only.

34.5.2.1 Hard lock

The hard lock is enabled by setting the SWT_CR[HLK] bit, which can only be cleared by a reset.

34.5.2.2 Soft lock

The soft lock is enabled by setting the SWT_CR[SLK] bit and is cleared by writing the unlock sequence to the service register.

34.5.3 Unlock sequence

The unlock sequence is a write of 0xC520 followed by a write of 0xD928 to the SWT_SR[WSC] field. There is no timing requirement between the two writes. The unlock sequence logic ignores service sequence writes and recognizes the 0xC520, 0xD928 sequence regardless of previous writes. The unlock sequence can be written at any time and does not require the SWT_CR[WEN] bit to be set.

34.5.4 Servicing operations

When enabled, the SWT requires periodic execution of a servicing operation that is determined by the SWT_CR[SMD] field. Properly servicing the watchdog loads the internal down counter with the time-out period. The servicing modes are:

- fixed service sequence
- keyed service sequence

34.5.4.1 Fixed service sequence mode

If the SWT_CR[SMD] field is 00b, the fixed service sequence mode is selected, which requires writing 0xA602, then 0xB480 to the SWT_SR[WSC] field to service the watchdog. There is no timing requirement between the two writes and the service sequence logic ignores unlock sequence writes.

34.5.4.2 Keyed service sequence mode

If the SWT_CR[SMD] field is 01b, then the keyed service sequence mode is selected, which requires writing two pseudorandom keys to the SWT_SR[WSC] field to service the watchdog. The key values are determined by the pseudorandom key generator defined by the equation in the following figure. This algorithm will generate a sequence of 2^{16} different key values before repeating. The state of the key generator is held in the SWT_SK register. For example, if SWT_SK[SK] is 0x0100, then the service sequence keys are 0x1103, 0x2136. In this mode, each time a valid key is written to the SWT_SR register, the SWT_SK register is updated. So, after servicing the watchdog by writing 0x1103 and then 0x2136 to the SWT_SR[WSC] field, SWT_SK[SK] is 0x2136 and the next key sequence is 0x3499, 0x7E2C.

$$SK_{n+1} = (17 \times SK_n + 3) \bmod 2^{16}$$

Figure 34-8. Pseudorandom Key Generator

34.5.4.3 Window mode

If window mode is enabled (SWT_CR[WND] bit is 1), the service sequence must be applied in the last part of the time-out period defined by SWT_WN.

- The window is open when the down counter is less than the value in SWT_WN.
- Outside this window, service sequence writes are invalid accesses and generate a bus error or reset request depending on the value of SWT_CR[RIA].

For example, if SWT_TO is set to 5000 and SWT_WN is set to 1000, the service sequence must be applied in the last 20% of the time-out period.

Synchronization logic in the watchdog design slightly delays when the window opens. This delay could be up to 3 system clock cycles plus 4 counter clock cycles.

34.5.5 Time-out

The Interrupt Then Reset bit (SWT_CR[ITR]) controls the SWT's action taken when a time-out occurs.

- If the SWT_CR[ITR] bit is 0, the SWT generates a reset request immediately on any time-out.
- If the SWT_CR[ITR] bit is 1, an initial time-out causes the SWT to generate an interrupt and load the down counter with the time-out period.
 - The interrupt is indicated by the Time-out Interrupt Flag (SWT_IR[TIF]).
 - Clear the interrupt by writing 1 to SWT_IR[TIF].

If the service sequence is not written before the second consecutive time-out, the SWT generates a reset request.

34.5.6 Initialization

All registers should be initialized before setting the SWT_CR[WEN] bit to enable the watchdog. Registers can be initialized in any sequence.

Chapter 35

Periodic Interrupt Timer (PIT-RTI)

35.1 Chip-specific PIT information

This chip has one instance of the PIT.

PIT channels can be used to trigger periodic ADC conversions via EQADC Command FIFOs. For more information, see:

- [eQADC Command FIFO Trigger Source Registers \(SIU_ISEL4\) to eQADC Command FIFO Trigger Source Registers \(SIU_ISEL7\) and eQADC Advance Trigger Source Register \(SIU_ISEL9\)](#)
- [SIU Trigger Burst Generators \(TBG\)](#)

35.2 Introduction

The PIT module is an array of timers that can be used to raise interrupts and triggers. It includes a dedicated Real Time Interrupt Timer (RTI), which runs on a separate clock and can be used for system wakeup

35.2.1 Block diagram

The following figure shows the block diagram of the PIT-RTI module.

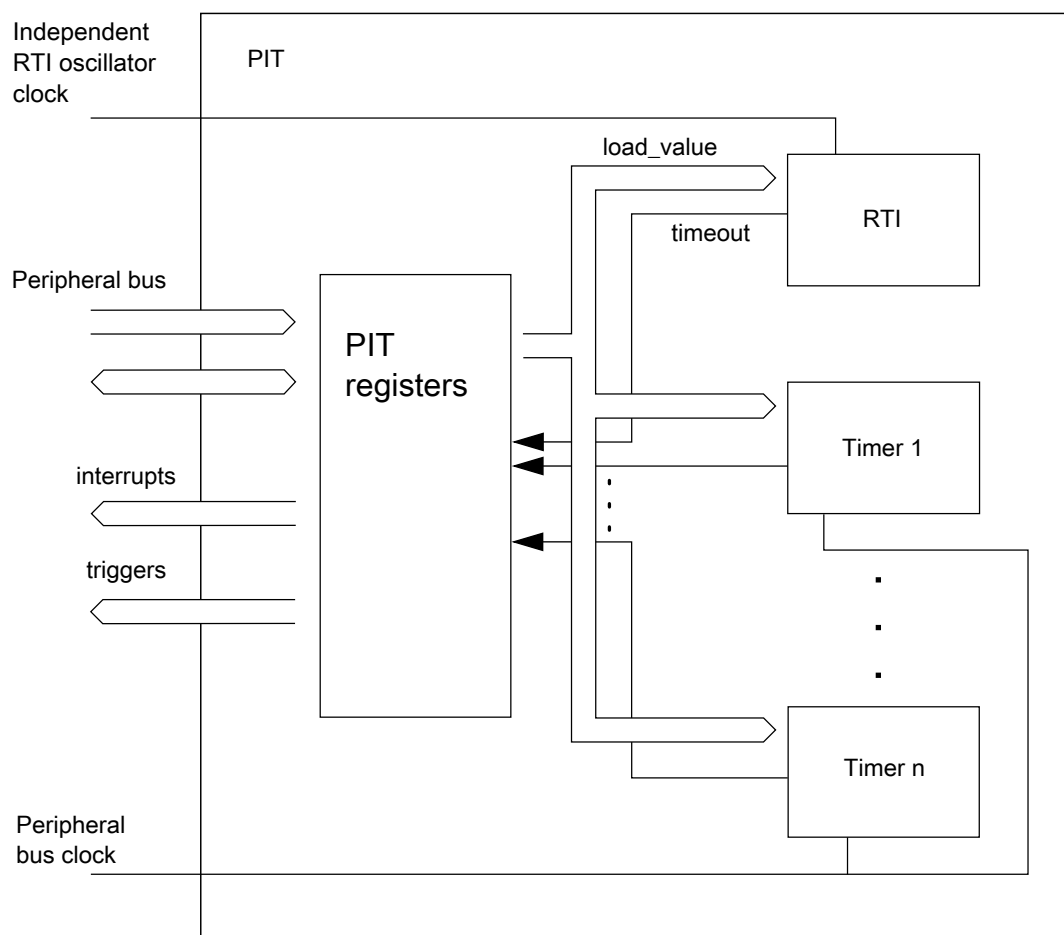


Figure 35-1. Block diagram of the PIT-RTI

NOTE

See the chip-specific PIT information for the number of PIT channels used in this MCU.

35.2.2 Features

The main features of this block are:

- One RTI (Real-Time Interrupt) timer to wakeup the CPU
- Ability of timers to generate trigger pulses
- Ability of timers to generate interrupts
- Maskable interrupts
- Option to raise RTI interrupt, even when the bus clock is switched off

- Power saving with a separate input clock for the RTI timer. All other timers share one common core clock.
- Independent timeout periods for each timer

35.3 Signal description

The PIT module has no external pins.

35.4 Memory map/register description

This section provides a detailed description of all registers accessible in the PIT_RTIC module.

- Reserved registers will read as 0, writes will have no effect.
- See the chip-specific PIT information for the number of PIT channels used in this MCU.
- The RTI registers should be programmed only when the RTI clock is running.

PIT memory map

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
0	PIT Module Control Register (PIT_MCR)	32	R/W	0000_0000h	35.4.1/1370
E0	PIT Upper Lifetime Timer Register (PIT_LTMR64H)	32	R	0000_0000h	35.4.2/1371
E4	PIT Lower Lifetime Timer Register (PIT_LTMR64L)	32	R	0000_0000h	35.4.3/1372
F0	Timer Load Value Register (PIT_RTIL_DVAL)	32	R/W	0000_0000h	35.4.4/1372
F4	Current Timer Value Register (PIT_RTIL_CVAL)	32	R	0000_0000h	35.4.5/1373
F8	Timer Control Register (PIT_RTIL_TCTRL)	32	R/W	0000_0000h	35.4.6/1373
FC	Timer Flag Register (PIT_RTIL_TFLG)	32	R/W	0000_0000h	35.4.7/1374
100	Timer Load Value Register (PIT_LDVAL0)	32	R/W	0000_0000h	35.4.8/1375
104	Current Timer Value Register (PIT_CVAL0)	32	R	0000_0000h	35.4.9/1375
108	Timer Control Register (PIT_TCTRL0)	32	R/W	0000_0000h	35.4.10/1376
10C	Timer Flag Register (PIT_TFLG0)	32	R/W	0000_0000h	35.4.11/1377
110	Timer Load Value Register (PIT_LDVAL1)	32	R/W	0000_0000h	35.4.8/1375
114	Current Timer Value Register (PIT_CVAL1)	32	R	0000_0000h	35.4.9/1375
118	Timer Control Register (PIT_TCTRL1)	32	R/W	0000_0000h	35.4.10/1376

Table continues on the next page...

PIT memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
11C	Timer Flag Register (PIT_TFLG1)	32	R/W	0000_0000h	35.4.11/1377
120	Timer Load Value Register (PIT_LDVAL2)	32	R/W	0000_0000h	35.4.8/1375
124	Current Timer Value Register (PIT_CVAL2)	32	R	0000_0000h	35.4.9/1375
128	Timer Control Register (PIT_TCTRL2)	32	R/W	0000_0000h	35.4.10/1376
12C	Timer Flag Register (PIT_TFLG2)	32	R/W	0000_0000h	35.4.11/1377
130	Timer Load Value Register (PIT_LDVAL3)	32	R/W	0000_0000h	35.4.8/1375
134	Current Timer Value Register (PIT_CVAL3)	32	R	0000_0000h	35.4.9/1375
138	Timer Control Register (PIT_TCTRL3)	32	R/W	0000_0000h	35.4.10/1376
13C	Timer Flag Register (PIT_TFLG3)	32	R/W	0000_0000h	35.4.11/1377

35.4.1 PIT Module Control Register (PIT_MCR)

This register enables or disables the PIT timer clocks and controls the timers when the PIT enters the Debug mode.

Access: User read/write

Address: 0h base + 0h offset = 0h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15															
R	0																														
W																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0															
Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31															
R	0																												MDIS_RTI	MDIS	FRZ
W																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0															

PIT_MCR field descriptions

Field	Description
0–28 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

Table continues on the next page...

PIT_MCR field descriptions (continued)

Field	Description
29 MDIS_RTI	Module Disable - RTI section This is used to disable the RTI timer. This bit must be enabled before any RTI setup is done. 0 Clock for RTI Timers is enabled 1 Clock for RTI Timers is disabled
30 MDIS	Module Disable - (PIT section) Disables the standard timers. The RTI timer is not affected by this field. This field must be enabled before any other setup is done. 0 Clock for standard PIT timers is enabled. 1 Clock for standard PIT timers is disabled.
31 FRZ	Freeze Allows the timers to be stopped when the device enters the Debug mode. 0 Timers continue to run in Debug mode. 1 Timers are stopped in Debug mode.

35.4.2 PIT Upper Lifetime Timer Register (PIT_LTMR64H)

This register is intended for applications that chain timer 0 and timer 1 to build a 64-bit lifetimer.

Access: User read only

Address: 0h base + E0h offset = E0h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	LTH																															
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	

PIT_LTMR64H field descriptions

Field	Description
0–31 LTH	Life Timer value Shows the timer value of timer 1. If this register is read at a time t1, LTMR64L shows the value of timer 0 at time t1.

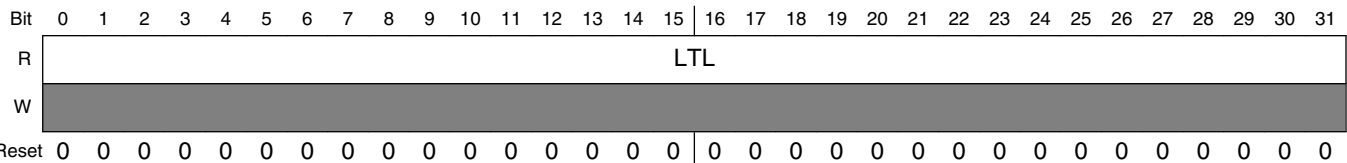
35.4.3 PIT Lower Lifetime Timer Register (PIT_LTMR64L)

This register is intended for applications that chain timer 0 and timer 1 to build a 64-bit lifetimer.

To use LTMR64H and LTMR64L, timer 0 and timer 1 need to be chained. To obtain the correct value, first read LTMR64H and then LTMR64L. LTMR64H will have the value of CVAL1 at the time of the first access, LTMR64L will have the value of CVAL0 at the time of the first access, therefore the application does not need to worry about carry-over effects of the running counter.

Access: User read only

Address: 0h base + E4h offset = E4h



PIT_LTMR64L field descriptions

Field	Description
0–31 LTL	Life Timer value Shows the value of timer 0 at the time LTMR64H was last read. It will only update if LTMR64H is read.

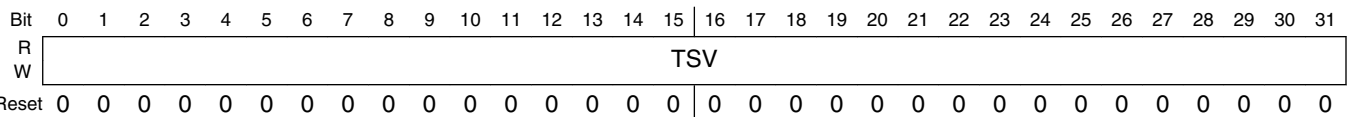
35.4.4 Timer Load Value Register (PIT_RTI_LDVAL)

These registers select the timeout period for the timer interrupts.

In the case of the RTI, it will take several cycles until this value is synchronized into the RTI clock domain. For all other timers the value change is visible immediately. The synchronisation mechanism allows 0 wait states in this case.

Access: User read/write

Address: 0h base + F0h offset = F0h



PIT_RTI_LDVAL field descriptions

Field	Description
0–31 TSV	Timer Start Value Sets the timer start value. The timer will count down until it reaches 0, then it will generate an interrupt and load this register value again. Writing a new value to this register will not restart the timer; instead the value will be loaded after the timer expires. To abort the current cycle and start a timer period with the new value, the timer must be disabled and enabled again. NOTE: The RTI timer must not be set to a value lower than 32 cycles, otherwise interrupts may be lost, as it takes several cycles to clear the RTI interrupt. For the other timers, this limit does not apply, however there will be practical limits, since the processor will require several cycles to service an interrupt.

35.4.5 Current Timer Value Register (PIT_RTI_CVAL)

These registers indicate the current timer position.

For the RTI timers, they show a value which is several cycles old, since it originates from a potentially different clock domain.

Access: User read only

Address: 0h base + F4h offset = F4h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	TVL																															
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	

PIT_RTI_CVAL field descriptions

Field	Description
0–31 TVL	Current Timer Value Represents the current timer value, if the timer is enabled. NOTE: - If the timer is disabled, do not use this field as its value is unreliable. - The timer uses a downcounter. The timer values are frozen in Debug mode if MCR[FRZ] is set.

35.4.6 Timer Control Register (PIT_RTI_TCTRL)

These registers contain the control bits for each timer.

Access: User read/write

Memory map/register description

Address: 0h base + F8h offset = F8h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0															TIE
W																TEN
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

PIT_RTI_TCTRL field descriptions

Field	Description
0–29 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
30 TIE	Timer Interrupt Enable When an interrupt is pending or, RTI_TFLG[TIF] is set, enabling the interrupt will immediately cause an interrupt event. To avoid this, the associated RTI_TFLG[TIF] flag must be cleared first. 0 Interrupt requests from the RTI Timer are disabled. 1 Interrupt will be requested whenever TIF is set.
31 TEN	Timer Enable Bit This bit enables or disables the timer. 0 RTI Timer is disabled. 1 RTI Timer is active.

35.4.7 Timer Flag Register (PIT_RTI_TFLG)

These registers hold the PIT interrupt flags.

Access: User read/write

Address: 0h base + FCh offset = FCh

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0															TIF
W																w1c
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

PIT_RTI_TFLG field descriptions

Field	Description
0–30 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
31 TIF	Timer Interrupt Flag. TIF is set to 1 at the end of the timer period. This flag can be cleared only by writing it with 1 . Writing 0 has no effect. If enabled (TIE = 1), TIF causes an interrupt request. 0 Time-out has not yet occurred. 1 Time-out has occurred.

35.4.8 Timer Load Value Register (PIT_LDVALn)

These registers select the timeout period for the timer interrupts.

Access: User read/write

Address: 0h base + 100h offset + (16d × i), where i=0d to 3d

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R																																
W	TSV																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	

PIT_LDVALn field descriptions

Field	Description
0–31 TSV	Timer Start Value Sets the timer start value. The timer will count down until it reaches 0, then it will generate an interrupt and load this register value again. Writing a new value to this register will not restart the timer; instead the value will be loaded after the timer expires. To abort the current cycle and start a timer period with the new value, the timer must be disabled and enabled again.

35.4.9 Current Timer Value Register (PIT_CVALn)

These registers indicate the current timer position.

Access: User read only

Address: 0h base + 104h offset + (16d × i), where i=0d to 3d

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R																	TVL															
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

PIT_CVAL_n field descriptions

Field	Description
0–31 TVL	Current Timer Value Represents the current timer value, if the timer is enabled. NOTE: - If the timer is disabled, do not use this field as its value is unreliable. - The timer uses a downcounter. The timer values are frozen in Debug mode if MCR[FRZ] is set.

35.4.10 Timer Control Register (PIT_TCTRL_n)

These registers contain the control bits for each timer.

Access: User read/write

Address: 0h base + 108h offset + (16d × i), where i=0d to 3d

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31			
R	0																		
W														CHN		TIE		TEN	
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0			

PIT_TCTRL_n field descriptions

Field	Description
0–28 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
29 CHN	Chain Mode When activated, Timer n-1 needs to expire before timer n can decrement by 1. Timer 0 cannot be chained. The RTI timer cannot be chained. 0 Timer is not chained. 1 Timer is chained to previous timer. For example, for Channel 2, if this field is set, Timer 2 is chained to Timer 1.
30 TIE	Timer Interrupt Enable When an interrupt is pending, or, TFLG_n[TIF] is set, enabling the interrupt will immediately cause an interrupt event. To avoid this, the associated TFLG_n[TIF] must be cleared first. 0 Interrupt requests from Timer n are disabled. 1 Interrupt will be requested whenever TIF is set.
31 TEN	Timer Enable Enables or disables the timer.

Table continues on the next page...

PIT_TCTRLn field descriptions (continued)

Field	Description
0	Timer n is disabled.
1	Timer n is enabled.

35.4.11 Timer Flag Register (PIT_TFLGn)

These registers hold the PIT interrupt flags.

Access: User read/write

Address: 0h base + 10Ch offset + (16d × i), where i=0d to 3d

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15															
R	0																														
W																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0															
Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31															
R	0																														TIF
W																															w1c
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0															

PIT_TFLGn field descriptions

Field	Description
0–30 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
31 TIF	Timer Interrupt Flag Sets to 1 at the end of the timer period. Writing 1 to this flag clears it. Writing 0 has no effect. If enabled, or, when TCTRLn[TIE] = 1, TIF causes an interrupt request. 0 Timeout has not yet occurred. 1 Timeout has occurred.

35.5 Functional description

This section provides the functional description of the module.

35.5.1 General operation

This section gives detailed information on the internal operation of the module. Each timer can be used to generate trigger pulses and interrupts. Each interrupt is available on a separate interrupt line. Additionally the RTI timer can be used to wake up the processor.

35.5.1.1 Timers / RTI

The timers generate triggers at periodic intervals, when enabled. The timers load the start values as specified in their LDVAL registers, count down to 0 and then load the respective start value again. Each time a timer reaches 0, it will generate a trigger pulse and set the interrupt flag.

All interrupts can be enabled or masked by setting TCTRLn[TIE]. A new interrupt can be generated only after the previous one is cleared. Since in the case of the RTI, clearing the interrupt crosses clock domains, a minimum load value of 32 must be maintained.

If desired, the current counter value of the timer can be read via the CVAL registers. The value of the RTI counter can be delayed considerably, as it is synchronized to the bus clock from the RTI clock domain.

The counter period can be restarted, by first disabling, and then enabling the timer with TCTRLn[TEN]. See the following figure.

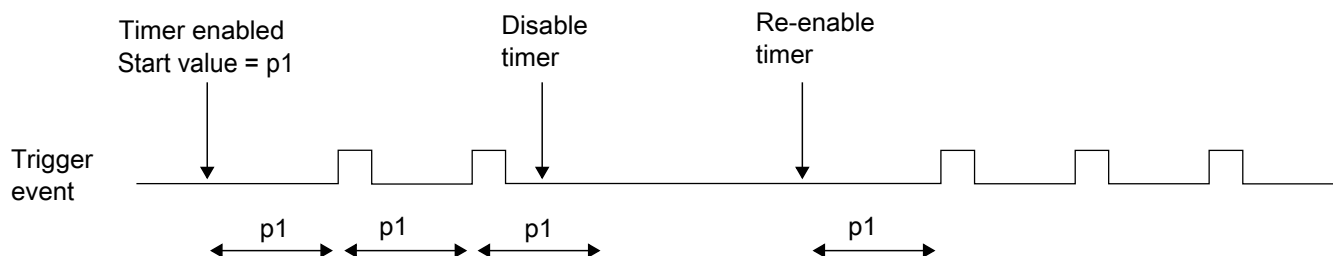


Figure 35-29. Stopping and starting a timer

The counter period of a running timer can be modified, by first disabling the timer, setting a new load value, and then enabling the timer again. See the following figure.

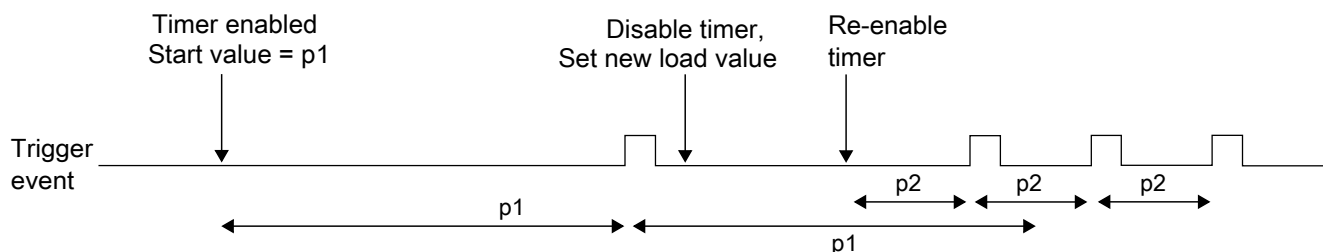


Figure 35-30. Modifying running timer period

It is also possible to change the counter period without restarting the timer by writing LDVAL with the new load value. This value will then be loaded after the next trigger event. See the following figure.

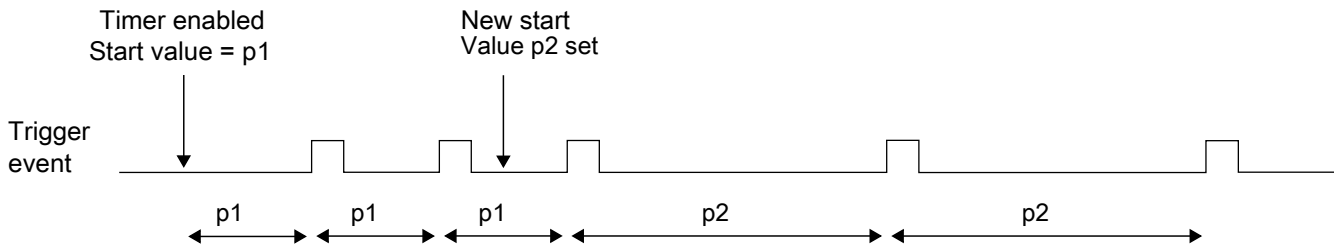


Figure 35-31. Dynamically setting a new load value

35.5.1.2 Debug mode

In Debug mode, the timers will be frozen based on MCR[FRZ]. This is intended to aid software development, allowing the developer to halt the processor, investigate the current state of the system, for example, the timer values, and then continue the operation.

35.5.2 Interrupts

All the timers support interrupt generation. The RTI is typically used for system wakeup, but can be used for interrupt generation as well. See the MCU specification for related vector addresses and priorities.

Timer interrupts can be enabled by setting TCTRLn[TIE]. TFLGn[TIF] are set to 1 when a timeout occurs on the associated timer, and are cleared to 0 by writing a 1 to the corresponding TFLGn[TIF].

The PIT_RTI generates a real time interrupt when the selected interrupt time period elapses. The RTI interrupt is disabled locally by setting TCTCRLn[TIE] to zero. The real time interrupt flag (TIF) is set to 1 when a timeout occurs, and is cleared by writing a 1 to the corresponding TFLGn[TIF]. The flag will be set regardless whether the interrupt is enabled.

The RTI can be used for periodic wakeup from a low-power mode. It can also be used to generate a general-purpose interrupt.

35.5.3 Chained timers

When a timer has chain mode enabled, it will only count after the previous timer has expired. So if timer n-1 has counted down to 0, counter n will decrement the value by one. This allows to chain some of the timers together to form a longer timer. The first timer (timer 0) cannot be chained to any other timer.

35.6 Initialization and application information

In the example configuration:

- The PIT clock has a frequency of 50 MHz.
- The RTI clock has a frequency of 10 MHz.
- The RTI creates a wakeup interrupt every 500 ms.
- Timer 1 creates an interrupt every 5.12 ms.
- Timer 3 creates a trigger event every 30 ms.

The PIT module must be activated by writing a 0 to MCR[MDIS].

The 50 MHz clock frequency equates to a clock period of 20 ns and the 10 MHz frequency equates to a clock period of 100 ns. Therefore, the RTI timer needs to trigger every $500 \text{ ms} / 100 \text{ ns} = 5,000,000$ cycles. Timer 1 needs to trigger every $5.12 \text{ ms} / 20 \text{ ns} = 256,000$ cycles and Timer 3 every $30 \text{ ms} / 20 \text{ ns} = 1,500,000$ cycles. The value for the LDVAL register trigger is calculated as:

$\text{LDVAL trigger} = (\text{period} / \text{clock period}) - 1$

This means RTI LDVAL will be written with 004C4B3F hex; LDVAL1 and LDVAL3 must be written with 0x0003E7FF and 0x0016E35F respectively.

To generate the wakeup interrupt, the interrupt line must be enabled by writing 1 to the RTI TIE bit in the TCTRL register. To start the RTI, the TEN in the RTI TCTRL register must also be set.

The interrupt for Timer 1 is enabled by setting TCTRL1[TIE]. The timer is started by writing 1 to TCTRL1[TEN].

Timer 3 shall be used only for triggering. Therefore, Timer 3 is started by writing a 1 to TCTRL3[TEN]. TCTRL3[TIE] stays at 0.

The following example code matches the described setup:


```
// turn on PIT
PIT_MCR = 0x00;

// RTI
PIT_RTI_LDVAL = 0x004C4B3F; // setup RTI for 5000000 cycles
PIT_RTI_TCTRL = PIT_TIE; // let RTI generate interrupts
PIT_RTI_TCTRL |= PIT_TEN; // start RTI

// Timer 1
PIT_LDVAL1 = 0x0003E7FF; // setup timer 1 for 256000 cycles
PIT_TCTRL1 = TIE; // enable Timer 1 interrupts
PIT_TCTRL1 |= TEN; // start Timer 1

// Timer 3
PIT_LDVAL3 = 0x0016E35F; // setup timer 3 for 1500000 cycles
PIT_TCTRL3 |= TEN; // start Timer 3
```

35.7 Example configuration for chained timers

In the example configuration:

- The PIT clock has a frequency of 100 MHz.
- Timers 1 and 2 are available.
- An interrupt shall be raised every 1 minute.

The PIT module needs to be activated by writing a 0 to MCR[MDIS].

The 100 MHz clock frequency equates to a clock period of 10 ns, so the PIT needs to count for 6000 million cycles, which is more than a single timer can do. So, Timer 1 is set up to trigger every 6 s (600 million cycles). Timer 2 is chained to Timer 1 and programmed to trigger 10 times.

The value for the LDVAL register trigger is calculated as number of cycles-1, so LDVAL1 receives the value 0x23C345FF and LDVAL2 receives the value 0x00000009.

The interrupt for Timer 2 is enabled by setting TCTRL2[TIE], the Chain mode is activated by setting TCTRL2[CHN], and the timer is started by writing a 1 to TCTRL2[TEN]. TCTRL1[TEN] needs to be set, and TCTRL1[CHN] and TCTRL1[TIE] are cleared.

The following example code matches the described setup:

```
// turn on PIT
PIT_MCR = 0x00;

// Timer 2
```

Example configuration for the lifetime timer

```
PIT_LDVAL2 = 0x00000009; // setup Timer 2 for 10 counts
PIT_TCTRL2 = TIE; // enable Timer 2 interrupt
PIT_TCTRL2 |= CHN; // chain Timer 2 to Timer 1
PIT_TCTRL2 |= TEN; // start Timer 2

// Timer 1
PIT_LDVAL1 = 0x23C345FF; // setup Timer 1 for 600 000 000 cycles
PIT_TCTRL1 = TEN; // start Timer 1
```

35.8 Example configuration for the lifetime timer

To configure the lifetime timer, channels 0 and 1 need to be chained together.

First the PIT module needs to be activated by writing a 0 to the MDIS bit in the CTRL register, then the LDVAL registers need to be set to the maximum value.

The timer is a downcounter.

The following example code matches the described setup:

```
// turn on PIT
PIT_MCR = 0x00;

// Timer 1
PIT_LDVAL1 = 0xFFFFFFFF; // setup timer 1 for maximum counting period
PIT_TCTRL1 = 0x0; // disable timer 1 interrupts
PIT_TCTRL1 |= CHN; // chain timer 1 to timer 0
PIT_TCTRL1 |= TEN; // start timer 1

// Timer 0
PIT_LDVAL0 = 0xFFFFFFFF; // setup timer 0 for maximum counting period
PIT_TCTRL0 = TEN; // start timer 0
```

To access the lifetime, read first LTMR64H and then LTMR64L.

```
current_uptime = PIT_LTMR64H<<32;
current_uptime = current_uptime + PIT_LTMR64L;
```

Chapter 36

Enhanced Modular IO Subsystem (eMIOS)

36.1 Chip-specific eMIOS information

This has two eMIOS module instances: eMIOS_0 and eMIOS_1. The two modules are identical and located on different Peripheral Bridges.

36.1.1 eMIOS channels and counter buses

Each eMIOS module instance has 16 channels, for a total of 32 channels. The 32 channels are split across the two instances to support redundancy for safety applications.

The following table shows the mapping of the 32 eMIOS external balls to the instance channels.

Table 36-1. eMIOS channel mapping

eMIOS balls	Instance channels	
	eMIOS_0	eMIOS_1
0-7	0-7	—
8-15	—	0-7
16-23	16-23	—
24-31	—	16-23

For both module instances, each set of implemented channels drives a counter bus:

- Channels 0-7 drive counter bus B.
- Channels 16-23 drive counter bus D.

36.1.2 eMIOS unimplemented channels

For both module instances, channels 8-15 and 24-31 are not implemented. References to features and functionality related to these channels—including registers, register fields, and counter buses—do not apply to this chip.

36.1.3 eMIOS interaction with other modules

The eMIOS modules interface with eTPU, DSPI, and IGF modules.

36.1.3.1 eMIOS global time base synchronization with eTPU

The Global Time Base Enable (GTBE) signal output of the eMIOS module instances, which is controlled by the eMIOS's MCR[GTBE] field, is ORed together with the GTBE signal output from the two eTPU instances. The ORed signal is then fed back to the GTBE input of all eMIOS and eTPU instances.

These connections enable global time base synchronization through the following procedure:

1. Negate the GTBE field in all eMIOS and eTPU instances.
2. Configure eMIOS and eTPU channels.
3. Assert the GTBE field in one of the eMIOS or eTPU instances.

36.1.3.2 eMIOS and eTPU ODIS connections

The following table lists the Output Disable (ODIS) connections between eMIOS and eTPU modules. The eMIOS modules' Cn[ODISSL] fields for the applicable channels use these connections.

Table 36-2. eMIOS and eTPU ODIS connections

Source		Destination		Description
Instance	Pin	Instance	Pin	
EMIOS_0	emios_flag_out[16:19]	EMIOS_0	emios_odis[3:0]	Connection from EMIOS_0 Channel Flag Out numbers 16, 17, 18, 19 to EMIOS_0 ODIS bits 3, 2, 1, 0, respectively
EMIOS_1	emios_flag_out[16:19]	EMIOS_1	emios_odis[3:0]	Connection from EMIOS_1 Channel Flag Out numbers 16, 17, 18, 19 to EMIOS_1 ODIS bits 3, 2, 1, 0, respectively

Table continues on the next page...

Table 36-2. eMIOS and eTPU ODIS connections (continued)

Source		Destination		Description
Instance	Pin	Instance	Pin	
EMIOS_0	emios_flag_out[16:19]	ETPU_C	etpu_odis_1[3:0]	Connection from EMIOS_0 Channel Flag Out numbers 16, 17, 18, 19 to ETPU_C ODIS bits 3, 2, 1, 0, respectively
EMIOS_1	emios_flag_out[16:19]	ETPU_A	etpu_odis_1[3:0]	Connection from EMIOS_1 Channel Flag Out numbers 16, 17, 18, 19 to ETPU_A ODIS bits 3, 2, 1, 0, respectively
EMIOS_1	emios_flag_out[23:20]	ETPU_B	etpu_odis_2[3:0]	Connection from EMIOS_1 Channel Flag Out numbers 23, 22, 21, 20 to ETPU_B ODIS bits 3, 2, 1, 0, respectively

36.1.3.3 STAC server IDs

STAC server IDs are defined in [STAC server IDs](#).

36.1.3.4 eMIOS and DSPI connections

See [DSPI and eMIOS direct connections](#) and [DSPI connections with SIU, eTPU, and eMIOS modules](#).

36.1.3.5 eMIOS inputs with Glitch Filters

See [IGF interaction with other modules](#) for connections between eMIOS and Input Glitch Filters (IGFs).

36.1.3.6 eMIOS and eTPU signal passthrough from pins via IGFs

On this chip, the IGF modules mediate between some pins and the eMIOS and eTPU modules. After system reset, the IGF modules are in Configuration mode by default. This mode disables IGF output and prevents the signals from the pins from reaching the eMIOS and eTPU modules.

The signals affected by this situation appear in [IGF interaction with other modules](#).

To route an affected signal directly to an eMIOS or eTPU instance, program the applicable channel of the applicable IGF instance:

1. Ensure that IGF_MCRn[MDIS] is 0 and that the IGF channel is disabled: IGF_MCRn[FGEN] is 0.
2. Configure the rising and falling edges for bypass:
 - Rising edge: Write 00b to IGF_MCRn[RFM].
 - Falling edge: Write 00b to IGF_MCRn[FFM].
3. Enable the IGF channel: Write 1 to IGF_MCRn[FGEN].

At the end of this sequence, the programmed filter enters Normal mode and passes through the affected signal.

36.2 Introduction

36.2.1 Overview

The configurable enhanced Modular Input/Output Subsystem (eMIOS) provides functionality to generate or measure time events. Its overall architecture resembles that of the MIOS timer.

The predecessor MIOS timer block provides a framework in which a set of sublocks with different timer functions are assembled to attend to the specific needs of an SoC. The eMIOS builds on this concept by using a Unified Channel module that provides a superset of the functionality of all the individual MIOS channels, while also providing a consistent user interface. This allows more flexibility as each Unified Channel can be programmed for different functions in different applications of the SoC. In addition, eMIOS architecture allows the use of Dedicated Channels which perform specific functions not included in MIOS inheritance.

The block diagram of the configurable eMIOS block is shown below.

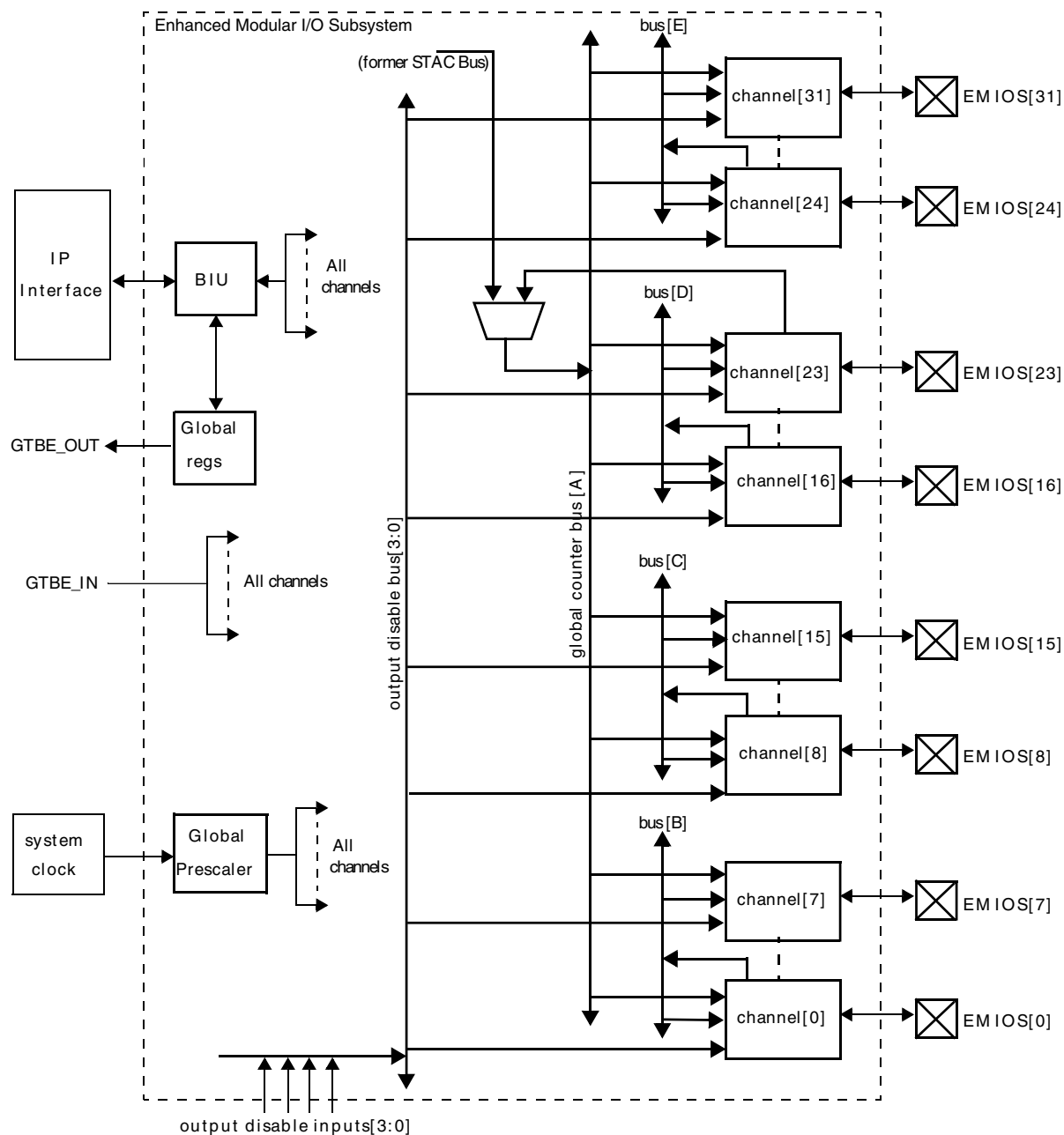


Figure 36-1. eMIOS full channel configuration using Unified Channels only (see Configuration section)

36.2.2 Features

The eMIOS has the following features:

- Up to 32 channels chosen among Unified or Dedicated Channels, not necessarily numbered in a continuous sequence. (See Configuration information)
- Data registers of either 8-, 16-, 24-, or 32-bit width. (See Configuration information)
- Counter buses B, C, D, and E can be driven by Unified Channels 0, 8, 16, and 24, respectively. (See Configuration information)
- Counter bus A can be driven by Unified Channel 23 or by the external shared timer bus(Red-Line Client).
- Each channel has its own timebase, alternative to the counter buses.
- One global prescaler
- One prescaler per channel (CP)
- Timebases shared through the counter buses
- One STAC Client (STAC)
- Control and status bits grouped in separated registers
- Synchronization among timebases
- Shadow FLAG register
- State of the Unified Channels can be frozen for debug purposes

36.2.3 Modes of operation

The Unified Channels can be configured to operate in the following modes:

- General-Purpose Input/Output (GPIO)
- Single-Action Input Capture (SAIC)
- Single-Action Output Compare (SAOC)
- Input Pulse-Width Measurement (IPWM)
- Input Period Measurement (IPM)
- Double-Action Output Compare (DAOC)
- Pulse/Edge Accumulation (PEA)
- Pulse/Edge Counting (PEC)
- Quadrature Decode (QDEC)
- Windowed Programmable Time Accumulation (WPTA)

- Modulus Counter (MC)
- Modulus Counter Buffered (MCB)
- Output Pulse-Width and Frequency Modulation (OPWFM)
- Output Pulse-Width and Frequency Modulation Buffered (OPWFMB)
- Center Aligned Output Pulse Width Modulation with Dead Time Insertion (OPWMC)
- Center Aligned Output Pulse Width Modulation with Dead Time Insertion Buffered (OPWMCB)
- Output Pulse-Width Modulation (OPWM)
- Output Pulse-Width Modulation Buffered (OPWMB)

These modes are described in [Unified Channel modes of operation](#).

Each channel can have a specific set of modes.

36.3 Memory map / register definition

36.3.1 Memory map

The overall address map organization is shown in the table below.

Whenever an access to either an absent register, absent channel, or reserved address space is performed, the transfer is terminated with an error.

Table 36-3. eMIOS Memory Map

eMIOS[n] Base Address	Description
0x000	Module Configuration register (EMIOSMCR)
0x003	
0x004	Global FLAG register (EMIOSGFLAG)
0x007	
0x008	Output Update Disable (EMIOSOUDIS)
0x00B	
0x00C	Reserved
0x00F	
0x01F	Reserved

Table continues on the next page...

Table 36-3. eMIOS Memory Map (continued)

eMIOS[n] Base Address	Description
0x020	Channel 0 to Channel 7
0x11F	
0x120	Channel 8 to Channel 15
0x21F	
0x220	Channel 16 to Channel 23
0x31F	
0x320	Channel 24 to Channel 31
0x41F	
0x420	Reserved
0xFFF	

36.3.1.1 Unified channel memory map

Addresses of Unified Channel registers are specified as offsets from the channel's base address, otherwise the eMIOS base address is used as reference.

The following table describes the Unified Channel memory map.

Table 36-4. Unified channel memory map

UC[n] Base Address	Description
UC[n]_BASE + 0x000	A register (EMIOSA[n])
UC[n]_BASE + 0x004	B register (EMIOSB[n])
UC[n]_BASE + 0x008	Counter register (EMIOSCNT[n])
UC[n]_BASE + 0x00C	Control register (EMIOSC[n])
UC[n]_BASE + 0x010	Status register (EMIOSS[n])
UC[n]_BASE + 0x014	Alternate A register (EMIOSALTA[n])
UC[n]_BASE + 0x018	Control 2 register (EMIOSC2[n])
UC[n]_BASE + 0x01C to UC[n]_BASE + 0x01F	Reserved

36.4 Control Memory Map and Registers

All control registers are 32 bits wide. This document illustrates the eMIOS with 24 Unified Channels and 24-bit wide data registers.

eMIOS memory map

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
0	eMIOS Module Configuration Register (eMIOS_MCR)	32	R/W	0000_0000h	36.4.1/1391
4	eMIOS Global FLAG Register (eMIOS_GFLAG)	32	R	0000_0000h	36.4.2/1393
8	eMIOS Output Update Disable register (eMIOS_OUDIS)	32	R/W	0000_0000h	36.4.3/1393

36.4.1 eMIOS Module Configuration Register (eMIOS_MCR)

The MCR register contains Global Control bits for the eMIOS block.

Address: 0h base + 0h offset = 0h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	Reserved	MDIS	FRZ	GTBE	ETB	GPREN	0						SRV			
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	GPRE								0							
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

eMIOS_MCR field descriptions

Field	Description
0 Reserved	Reserved. This field is reserved. Only write this field with the reset value.
1 MDIS	Module Disable bit. Puts the eMIOS into low power mode. The MDIS bit stops the clock of the block, except the access to registers EMIOSMCR and EMIOSOUDIS. 0 Clock is running 1 Enter low power mode
2 FRZ	Freeze bit. This bit enables the eMIOS to freeze the registers of the Unified Channels when Debug Mode is requested at MCU level. Each Unified Channel should have FREN bit set in order to enter freeze state. While in Freeze state, the eMIOS continues to operate to allow the MCU access to the Unified Channels registers. The Unified Channel will remain frozen until the FRZ bit is written to zero or the MCU exits Debug mode or the Unified Channel FREN bit is cleared.

Table continues on the next page...

eMIOS_MCR field descriptions (continued)

Field	Description
	0 Exit freeze state 1 Stops Unified Channels operation when in Debug mode and the FREN bit is set in the EMIOSC[n] register
3 GTBE	Global Time Base Enable bit. The GTBE bit is used to export a Global Time Base Enable from the module and provide a method to start time bases of several blocks simultaneously. NOTE: The Global Time Base Enable input pin controls the internal counters. When asserted, Internal counters are enabled. When negated, Internal counters disabled. 0 Global Time Base Enable Out signal negated 1 Global Time Base Enable Out signal asserted
4 ETB	External Time Base bit The ETB bit selects the time base source that drives counter bus[A]. 0 Counter bus[A] assigned to Unified Channel. 1 Counter bus[A] assigned to STAC.
5 GPREN	Global Prescaler Enable bit. The GPREN bit enables the prescaler counter. 0 Prescaler disabled (no clock) and prescaler counter is cleared 1 Prescaler enabled
6–11 Reserved	Reserved. This field is reserved. This read-only field is reserved and always has the value 0.
12–15 SRV	Server Time Slot bits. The SRV bits select the address of a specific STAC server to which the STAC is assigned.
16–23 GPRE	Global Prescaler bits. The GPRE[0:7] bits select the clock divider value for the global prescaler, as shown in the following table. GPRE values and corresponding divide ratio values are: 00000000 1 00000001 2 00000010 3 00000011 4 ----- - 11111110 255 11111111 256
24–31 Reserved	Reserved. This field is reserved. This read-only field is reserved and always has the value 0.

36.4.2 eMIOS Global FLAG Register (eMIOS_GFLAG)

GFLAG is a read-only register that groups the FLAG bits from all channels. Each bit relates to one channel.

Address: 0h base + 4h offset = 4h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0								F23_F16								0								F7_F0							
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	

eMIOS_GFLAG field descriptions

Field	Description
0–7 Reserved	Reserved. This field is reserved. This read-only field is reserved and always has the value 0.
8–15 F23_F16	Channel [n] mirrored FLAG bit. For Unified Channels these bits are mirrors of the FLAG bits in the EMIOSS[n] register.
16–23 Reserved	Reserved. This field is reserved. This read-only field is reserved and always has the value 0.
24–31 F7_F0	Channel [n] mirrored FLAG bit. For Unified Channels these bits are mirrors of the FLAG bits in the EMIOSS[n] register.

36.4.3 eMIOS Output Update Disable register (eMIOS_OUDIS)

Address: 0h base + 8h offset = 8h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0								OU23_OU16								0								OU7_OU0							
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	

eMIOS_OUDIS field descriptions

Field	Description
0–7 Reserved	Reserved. This field is reserved. This read-only field is reserved and always has the value 0.

Table continues on the next page...

eMIOS_OUDIS field descriptions (continued)

Field	Description
8–15 OU23_OU16	Channel [n] Output Update Disable bit. When running MC, MCB or an output mode, values are written to registers A2 and B2. OU[n] bits are used to disable transfers from registers A2 to A1 and B2 to B1. Each bit controls one channel. 0 Transfer enabled. Depending on the operation mode, transfer may occur immediately or in the next period. Unless stated otherwise, transfer occurs immediately. 1 Transfers disabled.
16–23 Reserved	Reserved. This field is reserved. This read-only field is reserved and always has the value 0.
24–31 OU7_OU0	Channel [n] Output Update Disable bit. When running MC, MCB or an output mode, values are written to registers A2 and B2. OU[n] bits are used to disable transfers from registers A2 to A1 and B2 to B1. Each bit controls one channel. 0 Transfer enabled. Depending on the operation mode, transfer may occur immediately or in the next period. Unless stated otherwise, transfer occurs immediately. 1 Transfers disabled.

36.5 Unified Channels Memory Map and Registers

The Unified Channel registers are as follows:

eMIOS memory map

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
20	eMIOS UC A register n (eMIOS_A0)	32	R/W	0000_0000h	36.5.1/1397
24	eMIOS UC B register n (eMIOS_B0)	32	R/W	0000_0000h	36.5.2/1398
28	eMIOS CNT register n (eMIOS_CNT0)	32	R/W	0000_0000h	36.5.3/1399
2C	eMIOS UC Control register n (eMIOS_C0)	32	R/W	0000_0000h	36.5.4/1400
30	eMIOS UC Status register n (eMIOS_S0)	32	R/W	See section	36.5.5/1406
34	eMIOS Alternate Address register n (eMIOS_ALTA0)	32	R/W	0000_0000h	36.5.6/1408
40	eMIOS UC A register n (eMIOS_A1)	32	R/W	0000_0000h	36.5.1/1397
44	eMIOS UC B register n (eMIOS_B1)	32	R/W	0000_0000h	36.5.2/1398
48	eMIOS CNT register n (eMIOS_CNT1)	32	R/W	0000_0000h	36.5.3/1399
4C	eMIOS UC Control register n (eMIOS_C1)	32	R/W	0000_0000h	36.5.4/1400
50	eMIOS UC Status register n (eMIOS_S1)	32	R/W	See section	36.5.5/1406
54	eMIOS Alternate Address register n (eMIOS_ALTA1)	32	R/W	0000_0000h	36.5.6/1408
60	eMIOS UC A register n (eMIOS_A2)	32	R/W	0000_0000h	36.5.1/1397

Table continues on the next page...

eMIOS memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
64	eMIOS UC B register n (eMIOS_B2)	32	R/W	0000_0000h	36.5.2/1398
68	eMIOS CNT register n (eMIOS_CNT2)	32	R/W	0000_0000h	36.5.3/1399
6C	eMIOS UC Control register n (eMIOS_C2)	32	R/W	0000_0000h	36.5.4/1400
70	eMIOS UC Status register n (eMIOS_S2)	32	R/W	See section	36.5.5/1406
74	eMIOS Alternate Address register n (eMIOS_ALTA2)	32	R/W	0000_0000h	36.5.6/1408
80	eMIOS UC A register n (eMIOS_A3)	32	R/W	0000_0000h	36.5.1/1397
84	eMIOS UC B register n (eMIOS_B3)	32	R/W	0000_0000h	36.5.2/1398
88	eMIOS CNT register n (eMIOS_CNT3)	32	R/W	0000_0000h	36.5.3/1399
8C	eMIOS UC Control register n (eMIOS_C3)	32	R/W	0000_0000h	36.5.4/1400
90	eMIOS UC Status register n (eMIOS_S3)	32	R/W	See section	36.5.5/1406
94	eMIOS Alternate Address register n (eMIOS_ALTA3)	32	R/W	0000_0000h	36.5.6/1408
A0	eMIOS UC A register n (eMIOS_A4)	32	R/W	0000_0000h	36.5.1/1397
A4	eMIOS UC B register n (eMIOS_B4)	32	R/W	0000_0000h	36.5.2/1398
A8	eMIOS CNT register n (eMIOS_CNT4)	32	R/W	0000_0000h	36.5.3/1399
AC	eMIOS UC Control register n (eMIOS_C4)	32	R/W	0000_0000h	36.5.4/1400
B0	eMIOS UC Status register n (eMIOS_S4)	32	R/W	See section	36.5.5/1406
B4	eMIOS Alternate Address register n (eMIOS_ALTA4)	32	R/W	0000_0000h	36.5.6/1408
C0	eMIOS UC A register n (eMIOS_A5)	32	R/W	0000_0000h	36.5.1/1397
C4	eMIOS UC B register n (eMIOS_B5)	32	R/W	0000_0000h	36.5.2/1398
C8	eMIOS CNT register n (eMIOS_CNT5)	32	R/W	0000_0000h	36.5.3/1399
CC	eMIOS UC Control register n (eMIOS_C5)	32	R/W	0000_0000h	36.5.4/1400
D0	eMIOS UC Status register n (eMIOS_S5)	32	R/W	See section	36.5.5/1406
D4	eMIOS Alternate Address register n (eMIOS_ALTA5)	32	R/W	0000_0000h	36.5.6/1408
E0	eMIOS UC A register n (eMIOS_A6)	32	R/W	0000_0000h	36.5.1/1397
E4	eMIOS UC B register n (eMIOS_B6)	32	R/W	0000_0000h	36.5.2/1398
E8	eMIOS CNT register n (eMIOS_CNT6)	32	R/W	0000_0000h	36.5.3/1399
EC	eMIOS UC Control register n (eMIOS_C6)	32	R/W	0000_0000h	36.5.4/1400
F0	eMIOS UC Status register n (eMIOS_S6)	32	R/W	See section	36.5.5/1406
F4	eMIOS Alternate Address register n (eMIOS_ALTA6)	32	R/W	0000_0000h	36.5.6/1408
100	eMIOS UC A register n (eMIOS_A7)	32	R/W	0000_0000h	36.5.1/1397
104	eMIOS UC B register n (eMIOS_B7)	32	R/W	0000_0000h	36.5.2/1398
108	eMIOS CNT register n (eMIOS_CNT7)	32	R/W	0000_0000h	36.5.3/1399
10C	eMIOS UC Control register n (eMIOS_C7)	32	R/W	0000_0000h	36.5.4/1400
110	eMIOS UC Status register n (eMIOS_S7)	32	R/W	See section	36.5.5/1406
114	eMIOS Alternate Address register n (eMIOS_ALTA7)	32	R/W	0000_0000h	36.5.6/1408
220	eMIOS UC A register n (eMIOS_A16)	32	R/W	0000_0000h	36.5.7/1409
224	eMIOS UC B register n (eMIOS_B16)	32	R/W	0000_0000h	36.5.8/1409
228	eMIOS CNT register n (eMIOS_CNT16)	32	R/W	0000_0000h	36.5.9/1410

Table continues on the next page...

eMIOS memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
22C	eMIOS UC Control register n (eMIOS_C16)	32	R/W	0000_0000h	36.5.10/1411
230	eMIOS UC Status register n (eMIOS_S16)	32	R/W	See section	36.5.11/1416
234	eMIOS Alternate Address register n (eMIOS_ALTA16)	32	R/W	0000_0000h	36.5.12/1418
240	eMIOS UC A register n (eMIOS_A17)	32	R/W	0000_0000h	36.5.7/1409
244	eMIOS UC B register n (eMIOS_B17)	32	R/W	0000_0000h	36.5.8/1409
248	eMIOS CNT register n (eMIOS_CNT17)	32	R/W	0000_0000h	36.5.9/1410
24C	eMIOS UC Control register n (eMIOS_C17)	32	R/W	0000_0000h	36.5.10/1411
250	eMIOS UC Status register n (eMIOS_S17)	32	R/W	See section	36.5.11/1416
254	eMIOS Alternate Address register n (eMIOS_ALTA17)	32	R/W	0000_0000h	36.5.12/1418
260	eMIOS UC A register n (eMIOS_A18)	32	R/W	0000_0000h	36.5.7/1409
264	eMIOS UC B register n (eMIOS_B18)	32	R/W	0000_0000h	36.5.8/1409
268	eMIOS CNT register n (eMIOS_CNT18)	32	R/W	0000_0000h	36.5.9/1410
26C	eMIOS UC Control register n (eMIOS_C18)	32	R/W	0000_0000h	36.5.10/1411
270	eMIOS UC Status register n (eMIOS_S18)	32	R/W	See section	36.5.11/1416
274	eMIOS Alternate Address register n (eMIOS_ALTA18)	32	R/W	0000_0000h	36.5.12/1418
280	eMIOS UC A register n (eMIOS_A19)	32	R/W	0000_0000h	36.5.7/1409
284	eMIOS UC B register n (eMIOS_B19)	32	R/W	0000_0000h	36.5.8/1409
288	eMIOS CNT register n (eMIOS_CNT19)	32	R/W	0000_0000h	36.5.9/1410
28C	eMIOS UC Control register n (eMIOS_C19)	32	R/W	0000_0000h	36.5.10/1411
290	eMIOS UC Status register n (eMIOS_S19)	32	R/W	See section	36.5.11/1416
294	eMIOS Alternate Address register n (eMIOS_ALTA19)	32	R/W	0000_0000h	36.5.12/1418
2A0	eMIOS UC A register n (eMIOS_A20)	32	R/W	0000_0000h	36.5.7/1409
2A4	eMIOS UC B register n (eMIOS_B20)	32	R/W	0000_0000h	36.5.8/1409
2A8	eMIOS CNT register n (eMIOS_CNT20)	32	R/W	0000_0000h	36.5.9/1410
2AC	eMIOS UC Control register n (eMIOS_C20)	32	R/W	0000_0000h	36.5.10/1411
2B0	eMIOS UC Status register n (eMIOS_S20)	32	R/W	See section	36.5.11/1416
2B4	eMIOS Alternate Address register n (eMIOS_ALTA20)	32	R/W	0000_0000h	36.5.12/1418
2C0	eMIOS UC A register n (eMIOS_A21)	32	R/W	0000_0000h	36.5.7/1409

Table continues on the next page...

eMIOS memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
2C4	eMIOS UC B register n (eMIOS_B21)	32	R/W	0000_0000h	36.5.8/1409
2C8	eMIOS CNT register n (eMIOS_CNT21)	32	R/W	0000_0000h	36.5.9/1410
2CC	eMIOS UC Control register n (eMIOS_C21)	32	R/W	0000_0000h	36.5.10/1411
2D0	eMIOS UC Status register n (eMIOS_S21)	32	R/W	See section	36.5.11/1416
2D4	eMIOS Alternate Address register n (eMIOS_ALTA21)	32	R/W	0000_0000h	36.5.12/1418
2E0	eMIOS UC A register n (eMIOS_A22)	32	R/W	0000_0000h	36.5.7/1409
2E4	eMIOS UC B register n (eMIOS_B22)	32	R/W	0000_0000h	36.5.8/1409
2E8	eMIOS CNT register n (eMIOS_CNT22)	32	R/W	0000_0000h	36.5.9/1410
2EC	eMIOS UC Control register n (eMIOS_C22)	32	R/W	0000_0000h	36.5.10/1411
2F0	eMIOS UC Status register n (eMIOS_S22)	32	R/W	See section	36.5.11/1416
2F4	eMIOS Alternate Address register n (eMIOS_ALTA22)	32	R/W	0000_0000h	36.5.12/1418
300	eMIOS UC A register n (eMIOS_A23)	32	R/W	0000_0000h	36.5.7/1409
304	eMIOS UC B register n (eMIOS_B23)	32	R/W	0000_0000h	36.5.8/1409
308	eMIOS CNT register n (eMIOS_CNT23)	32	R/W	0000_0000h	36.5.9/1410
30C	eMIOS UC Control register n (eMIOS_C23)	32	R/W	0000_0000h	36.5.10/1411
310	eMIOS UC Status register n (eMIOS_S23)	32	R/W	See section	36.5.11/1416
314	eMIOS Alternate Address register n (eMIOS_ALTA23)	32	R/W	0000_0000h	36.5.12/1418

36.5.1 eMIOS UC A register n (eMIOS_An)

This section describes the register for channels 0-7. For channels 16-23, see [eMIOS UC A register n \(eMIOS_An\)](#).

Address: 0h base + 20h offset + (32d × i), where i=0d to 7d

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0								A																							
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

eMIOS_An field descriptions

Field	Description
0–7 Reserved	Reserved. This field is reserved. This read-only field is reserved and always has the value 0.
8–31 A	Depending on the mode of operation, internal registers A1[0:23] or A2[0:23], used for matches and captures, can be assigned to address EMIOs_An. Both A1 and A2 are cleared by reset. The table in the EMIOs_Bn register description summarizes the EMIOs_An writing and reading accesses for all operation modes. Values not listed are reserved. For more information, see the “UC Modes of Operation” section.

36.5.2 eMIOS UC B register n (eMIOS_Bn)

This section describes the register for channels 0-7. For channels 16-23, see [eMIOS UC B register n \(eMIOS_Bn\)](#).

Depending on the mode of operation, internal registers B1[0:23] or B2[0:23] can be assigned to address EMIOs_Bn. Both B1 and B2 are cleared by reset. The table below summarizes the EMIOs_Bn writing and reading accesses for all operation modes. Values not listed are reserved. For more information see the “UC Modes of Operation” section.

Depending on the channel configuration, it may have an EMIOs_B register or not. EMIOs_B register is required for the following modes: DAOC, IPM, IPWM, OPWM, OPWMB, OPWFM, OPWFMB, OPWMC, OPWMCB, MC, MCB, PEA, PEC, WPTA.

The following table summarizes the EMIOs_An, EMIOs_Bn, and EMIOs_ALTAn writing and reading accesses for all operation modes. Values not listed are reserved. For more information, see the “UC Modes of Operation” section.

Table 36-13. EMIOs_An, EMIOs_Bn and EMIOs_ALTAn values assignment

Operation mode	Register access					
	Write	Read	Write	Read	Alt write	Alt read
GPIO	A1, A2	A1	B1, B2	B1	A2	A2
SAIC ¹	–	A2	B2	B2	–	–
SAOC	A2	A1	B2	B2	–	–
IPWM	–	A2	–	B1	–	–
IPM	–	A2	–	B1	–	–
DAOC	A2	A1	B2	B1	–	–
PEA	A1	A2	–	B1	–	–
PEC	A1	A1	B1	B1	–	A2
QDEC	A1	A1	B2	B2	–	–
WPTA	A1	A1	B1	B1	–	A2

Table continues on the next page...

Table 36-13. EMIOS_An, EMIOS_Bn and EMIOS_ALTAn values assignment (continued)

Operation mode	Register access					
	Write	Read	Write	Read	Alt write	Alt read
MC	A2	A1	B2	B2	–	–
OPWFM	A2	A1	B2	B1	–	–
OPWMC	A2	A1	B2	B1	–	–
OPWM	A2	A1	B2	B1	–	–
OPWMT	A2	A1	B2	B1	A2	A2
MCB	A2	A1	B2	B1	–	–
OPWFMB	A2	A1	B2	B1	–	–
OPWMCB	A2	A1	B2	B1	–	–
OPWMB	A2	A1	B2	B1	–	–

1. In these modes, the register EMIOS_Bn is not used, but B2 can be accessed.

Address: 0h base + 24h offset + (32d × i), where i=0d to 7d

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0								B																							
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

eMIOS_Bn field descriptions

Field	Description
0–7 Reserved	Reserved. This field is reserved. This read-only field is reserved and always has the value 0.
8–31 B	See text above register figure.

1. In these modes, the register EMIOS_Bn is not used, but B2 can be accessed.

36.5.3 eMIOS CNT register n (eMIOS_CNTn)

This section describes the register for channels 0-7. For channels 16-23, see [eMIOS CNT register n \(eMIOS_CNTn\)](#).

When GPIO mode is selected or the channel is frozen, the EMIOS_CNTn register is read/write. For all others modes, the EMIOS_CNTn is a read-only register. When entering some operation modes, this register is automatically cleared (refer to the “UC Modes of Operation” section for details).

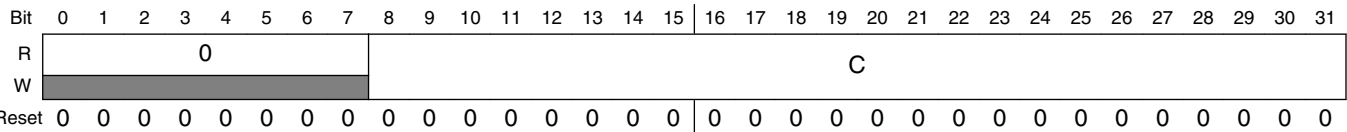
EMIOS_CNTn register is required for the following modes:

- OPWFM

- OPWFMB
- OPWMC
- OPWMCB
- PEA
- PEC
- WPTA
- QDEC
- MC
- MCB

For other modes, the internal counter is not used. Therefore, depending on the modes available in a channel, the internal counter may be absent.

Address: 0h base + 28h offset + (32d × i), where i=0d to 7d



eMIOS_CNTn field descriptions

Field	Description
0–7 Reserved	Reserved. This field is reserved. This read-only field is reserved and always has the value 0.
8–31 C	The C field contains the value of the internal counter.

36.5.4 eMIOS UC Control register n (eMIOS_Cn)

This section describes the register for channels 0-7. For channels 16-23, see [eMIOS UC Control register n \(eMIOS_Cn\)](#).

The UC Control register gathers several read/write control bits affecting the UC operation conditions.

NOTE

This register does not include a FORCMB bit for Channel Types A and C.

Address: 0h base + 2Ch offset + (32d × i), where i=0d to 7d

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R									0							0
W	FREN	ODIS	ODISSL		UCPRE		UCPREN	DMA			IF			FCK	FEN	
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0				0											
W			FORCMA	FORCMB		BSL		EDSEL	EDPOL							
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

eMIOS_Cn field descriptions

Field	Description						
0 FREN	Freeze Enable bit. The FREN bit, if set and validated by FRZ bit in EMIOS_MCR register allows the channel to enter freeze state, freezing all registers values when in debug mode and allowing the MCU to perform debug functions. 0 Normal operation 1 Freeze UC registers values						
1 ODIS	Output Disable bit. The ODIS bit allows disabling the output pin when running any of the output modes with the exception of GPIO mode. 0 The output pin operates normally 1 If the selected Output Disable Input signal is asserted, the output pin goes to EDPOL for OPWFMB and OPWMB modes and to the complement of EDPOL for other output modes, but the Unified Channel continues to operate normally, i.e., it continues to produce FLAG and matches. When the selected Output Disable Input signal is negated, the output pin operates normally.						
2–3 ODISSL	Output Disable select bits. The ODISSL[0:1] bits select one of the four output disable input signals, as shown in the following table. Table 36-23. UC ODISSL selection <table> <tr> <th>ODISSL[0:1]</th><th>input signal</th></tr> <tr> <td>00</td><td>Output Disable Input 0</td></tr> <tr> <td>01</td><td>Output Disable Input 1</td></tr> </table>	ODISSL[0:1]	input signal	00	Output Disable Input 0	01	Output Disable Input 1
ODISSL[0:1]	input signal						
00	Output Disable Input 0						
01	Output Disable Input 1						

Table continues on the next page...

eMIOS_Cn field descriptions (continued)

Field	Description										
	Table 36-23. UC ODISSL selection (continued) <table><tr><th>ODISSL[0:1]</th><th>input signal</th></tr><tr><td>10</td><td>Output Disable Input 2</td></tr><tr><td>11</td><td>Output Disable Input 3</td></tr></table>	ODISSL[0:1]	input signal	10	Output Disable Input 2	11	Output Disable Input 3				
ODISSL[0:1]	input signal										
10	Output Disable Input 2										
11	Output Disable Input 3										
4–5 UCPRE	Prescaler bits. The UCPRE[0:1] bits select the clock divider value for the internal prescaler of Unified Channel, as shown in the following table. Table 36-22. UC Internal Prescaler clock divider <table><tr><th>UCPRE[0:1]</th><th>Divide ratio</th></tr><tr><td>00</td><td>1</td></tr><tr><td>01</td><td>2</td></tr><tr><td>10</td><td>3</td></tr><tr><td>11</td><td>4</td></tr></table> NOTE: The UCPRE bits are mirrored in the two LSBs of the UCEXTPRE field in register EMIOS_Cn. Any write operations to UCPRE bits will affect the respective bits at EMIOS_C2n register.	UCPRE[0:1]	Divide ratio	00	1	01	2	10	3	11	4
UCPRE[0:1]	Divide ratio										
00	1										
01	2										
10	3										
11	4										
6 UCPREN	Prescaler Enable bit. The UCPREN bit enables the prescaler counter. 0 Prescaler disabled (no clock) 1 Prescaler enabled										
7 DMA	Direct Memory Access bit. The DMA bit selects if FLAG generation will be used as an interrupt or as a DMA request. 0 Flag/overflow assigned to Interrupt request. 1 Flag/overflow assigned to DMA request.										
8 Reserved	Reserved. This field is reserved. This read-only field is reserved and always has the value 0.										
9–12 IF	Input Filter bits. The IF[0:3] bits control the programmable input filter, selecting the minimum input pulse width that can pass through the filter, as shown in the following table. For output modes, these bits have no meaning. Table 36-21. UC Input Filter bits <table><tr><th>IF[0:3] ¹</th><th>Minimum input Pulse width [filter clock periods]</th></tr><tr><td>0000</td><td>bypassed²</td></tr><tr><td>0001</td><td>02</td></tr><tr><td>0010</td><td>04</td></tr></table>	IF[0:3] ¹	Minimum input Pulse width [filter clock periods]	0000	bypassed ²	0001	02	0010	04		
IF[0:3] ¹	Minimum input Pulse width [filter clock periods]										
0000	bypassed ²										
0001	02										
0010	04										

Table continues on the next page...

eMIOS_Cn field descriptions (continued)

Field	Description								
	Table 36-21. UC Input Filter bits (continued)								
	<table><tr><th>IF[0:3] ¹</th><th>Minimum input Pulse width [filter clock periods]</th></tr><tr><td>0100</td><td>08</td></tr><tr><td>1000</td><td>16</td></tr><tr><td>All others</td><td>Reserved</td></tr></table>	IF[0:3] ¹	Minimum input Pulse width [filter clock periods]	0100	08	1000	16	All others	Reserved
	IF[0:3] ¹	Minimum input Pulse width [filter clock periods]							
	0100	08							
	1000	16							
All others	Reserved								
1. Filter latency (the difference in time between the input and the response) is 3 clock edges. 2. The input signal is synchronized before arriving to the digital filter.									
13 FCK	Filter Clock select bit. The FCK bit selects the clock source for the programmable input filter. 0 Prescaled clock 1 eMIOS module clock								
14 FEN	FLAG Enable bit. The FEN bit allows the Unified Channel FLAG bit to generate an interrupt signal or a DMA request signal (the type of signal to be generated is defined by the DMA bit). 0 Disable (FLAG does not generate an interrupt request) 1 Enable (FLAG generates an interrupt request)								
15–17 Reserved	Reserved. This field is reserved. This read-only field is reserved and always has the value 0.								
18 FORCMA	Force Match A bit. For output modes, the FORCMA bit is equivalent to a successful comparison on comparator A (except that the FLAG bit is not set). This bit is cleared by reset and is always read as zero. This bit is valid for every output operation mode which uses comparator A, otherwise it has no effect. For input modes, the FORCMA bit is not used and writing to it has no effect. 0 Has no effect. 1 Force a match at comparator A								
19 FORCMB	Force Match B bit. For output modes, the FORCMB bit is equivalent to a successful comparison on comparator B (except that the FLAG bit is not set). This bit is cleared by reset and is always read as zero. This bit is valid for every output operation mode which uses comparator B, otherwise it has no effect. 0 Has not effect 1 Force a match at comparator B								
20 Reserved	Reserved. This field is reserved. This read-only field is reserved and always has the value 0.								
21–22 BSL	Bus Select bits.								

Table continues on the next page...

eMIOS_Cn field descriptions (continued)

Field	Description													
	The BSL[0:1] bits are used to select either one of the counter buses or the internal counter to be used by the Unified Channel , as shown in the following tables. Table 36-20. UC BSL bits <table> <tr> <th>BSL[0:1]</th><th>Selected bus¹</th></tr> <tr> <td>00</td><td>All channels: counter bus[A]</td></tr> <tr> <td rowspan="4">01</td><td>Channels 0 to 7: counter bus[B]</td></tr> <tr> <td>Channels 8 to 15: counter bus[C]</td></tr> <tr> <td>Channels 16 to 23: counter bus[D]</td></tr> <tr> <td>Channels 24 to 31: counter bus[E]</td></tr> <tr> <td>10</td><td>Reserved</td></tr> <tr> <td>11</td><td>All channels: internal counter</td></tr> </table> 1. Not all devices support all counter buses. See the Device Configuration chapter for details.	BSL[0:1]	Selected bus ¹	00	All channels: counter bus[A]	01	Channels 0 to 7: counter bus[B]	Channels 8 to 15: counter bus[C]	Channels 16 to 23: counter bus[D]	Channels 24 to 31: counter bus[E]	10	Reserved	11	All channels: internal counter
BSL[0:1]	Selected bus ¹													
00	All channels: counter bus[A]													
01	Channels 0 to 7: counter bus[B]													
	Channels 8 to 15: counter bus[C]													
	Channels 16 to 23: counter bus[D]													
	Channels 24 to 31: counter bus[E]													
10	Reserved													
11	All channels: internal counter													
23 EDSEL	Edge Selection bit. For input modes, the EDSEL bit selects whether the internal counter is triggered by both edges of a pulse or just by a single edge as defined by the EDPOL bit . When not shown in the mode of operation description, this bit has no effect. 0 – Single edge triggering defined by the EDPOL bit 1 – Both edges triggering{mts_ipi_slv_saic} For GPIO in mode, the EDSEL bit selects if a FLAG can be generated. 0 – A FLAG is generated as defined by the EDPOL bit 1 – No FLAG is generated For SAOC mode, the EDSEL bit selects the behavior of the output flip-flop at each match. 0 – The EDPOL value is transferred to the output flip-flop 1 – The output flip-flop is toggled													
24 EDPOL	Edge Polarity bit. For input modes (except QDEC mode), the EDPOL bit asserts which edge triggers either the internal counter or an input capture or a FLAG . When not shown in the mode of operation description, this bit has no effect. 0 – Trigger on a falling edge 1 – Trigger on a rising edge NOTE: UC[n-1] EDPOL bit selects which edge clocks the internal counter of UC[n] NOTE: In order to operate properly, EDPOL bit must contain the same value in UC[n] and UC[n-1] For output modes, the EDPOL bit is used to select the logic level on the output pin . 0 – A match on comparator A clears the output flip-flop, while a match on comparator B sets it 1 – A match on comparator A sets the output flip-flop, while a match on comparator B clears it													
25–31 MODE	Mode selection bits. The MODE bits select the mode of operation of the Unified Channel, as shown in the following table. Values not listed are reserved.													

Table continues on the next page...

eMIOS_Cn field descriptions

Field	Description																																																																				
	NOTE: If a reserved value is written to the mode bit field, the results are unpredictable. Table 36-19. UC MODE bits <table> <tr> <th>MODE[0:6]</th><th>Mode of operation¹</th></tr> <tr><td>0000000</td><td>General purpose Input/Output mode (input)</td></tr> <tr><td>0000001</td><td>General purpose Input/Output mode (output)</td></tr> <tr><td>0000010</td><td>Single Action Input Capture</td></tr> <tr><td>0000011</td><td>Single Action Output Compare</td></tr> <tr><td>0000100</td><td>Input Pulse Width Measurement</td></tr> <tr><td>0000101</td><td>Input Period Measurement</td></tr> <tr><td>0000110</td><td>Double Action Output compare (with FLAG set on B match)</td></tr> <tr><td>0000111</td><td>Double Action Output compare (with FLAG set on both match)</td></tr> <tr><td>0001000</td><td>Pulse/Edge Accumulation (continuous)</td></tr> <tr><td>0001001</td><td>Pulse/Edge Accumulation (single shot)</td></tr> <tr><td>0001010</td><td>Pulse/Edge Counting (continuous)</td></tr> <tr><td>0001011</td><td>Pulse/Edge Counting (single shot)</td></tr> <tr><td>0001100</td><td>Quadrature Decode (for count and direction encoders type)</td></tr> <tr><td>0001101</td><td>Quadrature Decode (for phase_A and phase_B encoders type)</td></tr> <tr><td>0001110</td><td>Windowed Programmable Time Accumulation</td></tr> <tr><td>0001111</td><td>Reserved</td></tr> <tr><td>001000b²</td><td>Modulus Counter (Up counter with clear on match start)</td></tr> <tr><td>001001b</td><td>Modulus Counter (Up counter with clear on match end)</td></tr> <tr><td>00101bb</td><td>Modulus Counter (Up/Down counter)</td></tr> <tr><td>00110b0</td><td>Output Pulse Width and Frequency Modulation (immediate update)</td></tr> <tr><td>00110b1</td><td>Output Pulse Width and Frequency Modulation (next period update)</td></tr> <tr><td>00111b0</td><td>Center Aligned Output Pulse Width Modulation (with trail edge dead-time)</td></tr> <tr><td>00111b1</td><td>Center Aligned Output Pulse Width Modulation (with lead edge dead-time)</td></tr> <tr><td>01000b0</td><td>Output Pulse Width Modulation (immediate update)</td></tr> <tr><td>01000b1</td><td>Output Pulse Width Modulation (next period update)</td></tr> <tr><td>0100100 through 0100101</td><td>Reserved</td></tr> <tr><td>0100110</td><td>Output Pulse Width Modulation with Trigger</td></tr> <tr><td>0100111 through 1001111</td><td>Reserved</td></tr> <tr><td>101000b</td><td>Modulus Counter Buffered (Up counter)</td></tr> <tr><td>101001b</td><td>Reserved</td></tr> <tr><td>10101bb</td><td>Modulus Counter Buffered (Up/Down counter)</td></tr> <tr><td>10110b0</td><td>Output Pulse Width and Frequency Modulation Buffered</td></tr> <tr><td>10110b1</td><td>Reserved</td></tr> </table>	MODE[0:6]	Mode of operation ¹	0000000	General purpose Input/Output mode (input)	0000001	General purpose Input/Output mode (output)	0000010	Single Action Input Capture	0000011	Single Action Output Compare	0000100	Input Pulse Width Measurement	0000101	Input Period Measurement	0000110	Double Action Output compare (with FLAG set on B match)	0000111	Double Action Output compare (with FLAG set on both match)	0001000	Pulse/Edge Accumulation (continuous)	0001001	Pulse/Edge Accumulation (single shot)	0001010	Pulse/Edge Counting (continuous)	0001011	Pulse/Edge Counting (single shot)	0001100	Quadrature Decode (for count and direction encoders type)	0001101	Quadrature Decode (for phase_A and phase_B encoders type)	0001110	Windowed Programmable Time Accumulation	0001111	Reserved	001000b ²	Modulus Counter (Up counter with clear on match start)	001001b	Modulus Counter (Up counter with clear on match end)	00101bb	Modulus Counter (Up/Down counter)	00110b0	Output Pulse Width and Frequency Modulation (immediate update)	00110b1	Output Pulse Width and Frequency Modulation (next period update)	00111b0	Center Aligned Output Pulse Width Modulation (with trail edge dead-time)	00111b1	Center Aligned Output Pulse Width Modulation (with lead edge dead-time)	01000b0	Output Pulse Width Modulation (immediate update)	01000b1	Output Pulse Width Modulation (next period update)	0100100 through 0100101	Reserved	0100110	Output Pulse Width Modulation with Trigger	0100111 through 1001111	Reserved	101000b	Modulus Counter Buffered (Up counter)	101001b	Reserved	10101bb	Modulus Counter Buffered (Up/Down counter)	10110b0	Output Pulse Width and Frequency Modulation Buffered	10110b1	Reserved
MODE[0:6]	Mode of operation ¹																																																																				
0000000	General purpose Input/Output mode (input)																																																																				
0000001	General purpose Input/Output mode (output)																																																																				
0000010	Single Action Input Capture																																																																				
0000011	Single Action Output Compare																																																																				
0000100	Input Pulse Width Measurement																																																																				
0000101	Input Period Measurement																																																																				
0000110	Double Action Output compare (with FLAG set on B match)																																																																				
0000111	Double Action Output compare (with FLAG set on both match)																																																																				
0001000	Pulse/Edge Accumulation (continuous)																																																																				
0001001	Pulse/Edge Accumulation (single shot)																																																																				
0001010	Pulse/Edge Counting (continuous)																																																																				
0001011	Pulse/Edge Counting (single shot)																																																																				
0001100	Quadrature Decode (for count and direction encoders type)																																																																				
0001101	Quadrature Decode (for phase_A and phase_B encoders type)																																																																				
0001110	Windowed Programmable Time Accumulation																																																																				
0001111	Reserved																																																																				
001000b ²	Modulus Counter (Up counter with clear on match start)																																																																				
001001b	Modulus Counter (Up counter with clear on match end)																																																																				
00101bb	Modulus Counter (Up/Down counter)																																																																				
00110b0	Output Pulse Width and Frequency Modulation (immediate update)																																																																				
00110b1	Output Pulse Width and Frequency Modulation (next period update)																																																																				
00111b0	Center Aligned Output Pulse Width Modulation (with trail edge dead-time)																																																																				
00111b1	Center Aligned Output Pulse Width Modulation (with lead edge dead-time)																																																																				
01000b0	Output Pulse Width Modulation (immediate update)																																																																				
01000b1	Output Pulse Width Modulation (next period update)																																																																				
0100100 through 0100101	Reserved																																																																				
0100110	Output Pulse Width Modulation with Trigger																																																																				
0100111 through 1001111	Reserved																																																																				
101000b	Modulus Counter Buffered (Up counter)																																																																				
101001b	Reserved																																																																				
10101bb	Modulus Counter Buffered (Up/Down counter)																																																																				
10110b0	Output Pulse Width and Frequency Modulation Buffered																																																																				
10110b1	Reserved																																																																				

eMIOS_Cn field descriptions (continued)

Field	Description	
	Table 36-19. UC MODE bits (continued)	
	MODE[0:6]	Mode of operation¹
	10111b0	Center Aligned Output Pulse Width Modulation Buffered (with trail edge dead-time)
	10111b1	Center Aligned Output Pulse Width Modulation Buffered (with lead edge dead-time)
	11000b0	Output Pulse Width Modulation Buffered
	1100001 through 1111111	Reserved
1. Not all modes of operation are supported. See the configuration information for details. 2. b = adjust parameters for the mode of operation. Refer to “UC Modes of Operation” section for details.		

1. Filter latency (the difference in time between the input and the response) is 3 clock edges.
2. The input signal is synchronized before arriving to the digital filter.
3. Not all devices support all counter buses. See the Device Configuration chapter for details.
4. Not all modes of operation are supported. See the configuration information for details.
5. b = adjust parameters for the mode of operation. Refer to “UC Modes of Operation” section for details.

36.5.5 eMIOS UC Status register n (eMIOS_Sn)

This section describes the register for channels 0-7. For channels 16-23, see [eMIOS UC Status register n \(eMIOS_Sn\)](#).

The UC Status register EMIOS_Sn gathers bits reflecting the status of the UC input/output signals and the overflow condition of the internal counter.

Address: 0h base + 30h offset + (32d × i), where i=0d to 7d

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	OVR	0														
W	w1c															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	OVFL	0												UCIN	UCOUT	FLAG
W	w1c															w1c
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	*	0	0

* Notes:

- UCIN field: Reset value is undefined.

eMIOS_*Sn* field descriptions

Field	Description
0 OVR	Overflow bit. The OVR bit indicates that FLAG generation occurred when the FLAG bit was already set . The OVR bit can be cleared either by clearing the FLAG bit or by writing a 1 to the OVR bit. 0 Overflow has not occurred 1 Overflow has occurred
1–15 Reserved	Reserved. This field is reserved. This read-only field is reserved and always has the value 0.
16 OVFL	Overflow bit. The OVFL bit indicates that an overflow has occurred in the internal counter. The OVFL bit must be cleared by software writing a 1 to it. 0 No overflow 1 An overflow had occurred
17–28 Reserved	Reserved. This field is reserved. This read-only field is reserved and always has the value 0.
29 UCIN	Unified Channel Input pin bit. The UCIN bit reflects the input pin state after being filtered and synchronized.
30 UCOUT	Unified Channel Output pin bit. The UCOUT bit reflects the output pin state .
31 FLAG	FLAG bit. The FLAG bit is set when an input capture or a match event in the comparators occurred. The FLAG bit must be cleared by writing a 1 to it.

Table continues on the next page...

eMIOS_Sn field descriptions (continued)

Field	Description
0	FLAG cleared
1	FLAG set event has occurred

36.5.6 eMIOS Alternate Address register n (eMIOS_ALTAn)

This section describes the register for channels 0-7. For channels 16-23, see [eMIOS Alternate Address register n \(eMIOS_ALTAn\)](#).

The eMIOS_ALTAn register provides an alternate address to access A2 channel registers in restricted modes (GPIO, PEC, WPTA) only. In modes where the function of eMIOS_ALTAn register is not specified, any write access is ignored and any read access returns unspecified data. If eMIOS_An register is used along with eMIOS_ALTAn, both A1 and A2 registers can be accessed in these modes. The table in the eMIOS_Bn register's description summarizes the EMIOS_ALTAn writing and reading accesses for all EMIOS_ALTAn operation modes. See the “General purpose Input/Output mode (GPIO) Mode” section, the “Pulse/Edge Counting (PEC) Mode” section, the “Windowed Programmable Time Accumulation (WPTA) Mode” section for a more detailed description of the use of eMIOS_ALTAn register.

Address: 0h base + 34h offset + (32d × i), where i=0d to 7d

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0								ALTA																							
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	

eMIOS_ALTAn field descriptions

Field	Description
0–7 Reserved	Reserved. This field is reserved. This read-only field is reserved and always has the value 0.
8–31 ALTA	See text above register figure.

36.5.7 eMIOS UC A register n (eMIOS_An)

This section describes the register for channels 16-23. For channels 0-7, see [eMIOS UC A register n \(eMIOS_An\)](#).

Address: 0h base + 220h offset + (32d × i), where i=0d to 7d

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0								A																							
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

eMIOS_An field descriptions

Field	Description
0–7 Reserved	Reserved. This field is reserved. This read-only field is reserved and always has the value 0.
8–31 A	Depending on the mode of operation, internal registers A1[0:23] or A2[0:23], used for matches and captures, can be assigned to address EMIOs_An. Both A1 and A2 are cleared by reset. The table in the EMIOs_Bn register description summarizes the EMIOs_An writing and reading accesses for all operation modes. Values not listed are reserved. For more information, see the “UC Modes of Operation” section.

36.5.8 eMIOS UC B register n (eMIOS_Bn)

This section describes the register for channels 16-23. In all other respects, this register is identical to the register for channels 0-7. See that register's description: [eMIOS UC B register n \(eMIOS_Bn\)](#).

Address: 0h base + 224h offset + (32d × i), where i=0d to 7d

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0								B																							
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

eMIOS_Bn field descriptions

Field	Description
0–7 Reserved	Reserved. This field is reserved. This read-only field is reserved and always has the value 0.

Table continues on the next page...

eMIOS_Bn field descriptions (continued)

Field	Description
8–31 B	See text above register figure.

36.5.9 eMIOS CNT register n (eMIOS_CNTn)

This section describes the register for channels 16-23. In all other respects, this register is identical to the register for channels 0-7. See that register's description: [eMIOS CNT register n \(eMIOS_CNTn\)](#).

Address: 0h base + 228h offset + (32d × i), where i=0d to 7d

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0								C																							
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	

eMIOS_CNTn field descriptions

Field	Description
0–7 Reserved	Reserved. This field is reserved. This read-only field is reserved and always has the value 0.
8–31 C	The C field contains the value of the internal counter.

36.5.10 eMIOS UC Control register n (eMIOS_Cn)

This section describes the register for channels 16-23. In all other respects, this register is identical to the register for channels 0-7. See that register's description: [eMIOS UC Control register n \(eMIOS_Cn\)](#).

Address: 0h base + 22Ch offset + (32d × i), where i=0d to 7d

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R									0							0
W	FREN	ODIS	ODISL		UCPRE		UCPREN	DMA			IF			FCK	FEN	
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0				0											
W			FORCMA	FORCMB			BSL	EDSEL	EDPOL							
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

eMIOS_Cn field descriptions

Field	Description
0 FREN	Freeze Enable bit. The FREN bit, if set and validated by FRZ bit in EMIOS_MCR register allows the channel to enter freeze state, freezing all registers values when in debug mode and allowing the MCU to perform debug functions. 0 Normal operation 1 Freeze UC registers values
1 ODIS	Output Disable bit. The ODIS bit allows disabling the output pin when running any of the output modes with the exception of GPIO mode.

Table continues on the next page...

eMIOS_Cn field descriptions (continued)

Field	Description										
0	The output pin operates normally										
1	If the selected Output Disable Input signal is asserted, the output pin goes to EDPOL for OPWFMB and OPWMB modes and to the complement of EDPOL for other output modes, but the Unified Channel continues to operate normally, i.e., it continues to produce FLAG and matches. When the selected Output Disable Input signal is negated, the output pin operates normally.										
2–3 ODISSL	Output Disable select bits. The ODISSL[0:1] bits select one of the four output disable input signals, as shown in the following table. Table 36-129. UC ODISSL selection <table> <tr> <th>ODISSL[0:1]</th><th>input signal</th></tr> <tr> <td>00</td><td>Output Disable Input 0</td></tr> <tr> <td>01</td><td>Output Disable Input 1</td></tr> <tr> <td>10</td><td>Output Disable Input 2</td></tr> <tr> <td>11</td><td>Output Disable Input 3</td></tr> </table>	ODISSL[0:1]	input signal	00	Output Disable Input 0	01	Output Disable Input 1	10	Output Disable Input 2	11	Output Disable Input 3
ODISSL[0:1]	input signal										
00	Output Disable Input 0										
01	Output Disable Input 1										
10	Output Disable Input 2										
11	Output Disable Input 3										
4–5 UCPRE	Prescaler bits. The UCPRE[0:1] bits select the clock divider value for the internal prescaler of Unified Channel, as shown in the following table. Table 36-128. UC Internal Prescaler clock divider <table> <tr> <th>UCPRE[0:1]</th><th>Divide ratio</th></tr> <tr> <td>00</td><td>1</td></tr> <tr> <td>01</td><td>2</td></tr> <tr> <td>10</td><td>3</td></tr> <tr> <td>11</td><td>4</td></tr> </table> NOTE: The UCPRE bits are mirrored in the two LSBs of the UCEXTPRE field in register EMIOS_Cn. Any write operations to UCPRE bits will affect the respective bits at EMIOS_C2n register.	UCPRE[0:1]	Divide ratio	00	1	01	2	10	3	11	4
UCPRE[0:1]	Divide ratio										
00	1										
01	2										
10	3										
11	4										
6 UCPREN	Prescaler Enable bit. The UCPREN bit enables the prescaler counter. 0 Prescaler disabled (no clock) 1 Prescaler enabled										
7 DMA	Direct Memory Access bit. The DMA bit selects if FLAG generation will be used as an interrupt or as a DMA request. 0 Flag/overflow assigned to Interrupt request. 1 Flag/overflow assigned to DMA request.										
8 Reserved	Reserved. This field is reserved. This read-only field is reserved and always has the value 0.										

Table continues on the next page...

eMIOS_Cn field descriptions (continued)

Field	Description														
9–12 IF	Input Filter bits. The IF[0:3] bits control the programmable input filter, selecting the minimum input pulse width that can pass through the filter, as shown in the following table. For output modes, these bits have no meaning. Table 36-127. UC Input Filter bits <table> <tr> <th>IF[0:3] ¹</th><th>Minimum input Pulse width [filter clock periods]</th></tr> <tr> <td>0000</td><td>bypassed²</td></tr> <tr> <td>0001</td><td>02</td></tr> <tr> <td>0010</td><td>04</td></tr> <tr> <td>0100</td><td>08</td></tr> <tr> <td>1000</td><td>16</td></tr> <tr> <td>All others</td><td>Reserved</td></tr> </table> 1. Filter latency (the difference in time between the input and the response) is 3 clock edges. 2. The input signal is synchronized before arriving to the digital filter.	IF[0:3] ¹	Minimum input Pulse width [filter clock periods]	0000	bypassed ²	0001	02	0010	04	0100	08	1000	16	All others	Reserved
IF[0:3] ¹	Minimum input Pulse width [filter clock periods]														
0000	bypassed ²														
0001	02														
0010	04														
0100	08														
1000	16														
All others	Reserved														
13 FCK	Filter Clock select bit. The FCK bit selects the clock source for the programmable input filter. 0 Prescaled clock 1 eMIOS module clock														
14 FEN	FLAG Enable bit. The FEN bit allows the Unified Channel FLAG bit to generate an interrupt signal or a DMA request signal (the type of signal to be generated is defined by the DMA bit). 0 Disable (FLAG does not generate an interrupt request) 1 Enable (FLAG generates an interrupt request)														
15–17 Reserved	Reserved. This field is reserved. This read-only field is reserved and always has the value 0.														
18 FORCMA	Force Match A bit. For output modes, the FORCMA bit is equivalent to a successful comparison on comparator A (except that the FLAG bit is not set). This bit is cleared by reset and is always read as zero. This bit is valid for every output operation mode which uses comparator A, otherwise it has no effect. For input modes, the FORCMA bit is not used and writing to it has no effect. 0 Has no effect. 1 Force a match at comparator A														
19 FORCMB	Force Match B bit. For output modes, the FORCMB bit is equivalent to a successful comparison on comparator B (except that the FLAG bit is not set). This bit is cleared by reset and is always read as zero. This bit is valid for every output operation mode which uses comparator B, otherwise it has no effect. 0 Has not effect 1 Force a match at comparator B														

Table continues on the next page...

eMIOS_Cn field descriptions (continued)

Field	Description													
20 Reserved	Reserved. This field is reserved. This read-only field is reserved and always has the value 0.													
21–22 BSL	Bus Select bits. The BSL[0:1] bits are used to select either one of the counter buses or the internal counter to be used by the Unified Channel , as shown in the following tables. Table 36-126. UC BSL bits <table> <tr> <th>BSL[0:1]</th><th>Selected bus¹</th></tr> <tr> <td>00</td><td>All channels: counter bus[A]</td></tr> <tr> <td rowspan="4">01</td><td>Channels 0 to 7: counter bus[B]</td></tr> <tr> <td>Channels 8 to 15: counter bus[C]</td></tr> <tr> <td>Channels 16 to 23: counter bus[D]</td></tr> <tr> <td>Channels 24 to 31: counter bus[E]</td></tr> <tr> <td>10</td><td>Reserved</td></tr> <tr> <td>11</td><td>All channels: internal counter</td></tr> </table> 1. Not all devices support all counter buses. See the Device Configuration chapter for details.	BSL[0:1]	Selected bus ¹	00	All channels: counter bus[A]	01	Channels 0 to 7: counter bus[B]	Channels 8 to 15: counter bus[C]	Channels 16 to 23: counter bus[D]	Channels 24 to 31: counter bus[E]	10	Reserved	11	All channels: internal counter
BSL[0:1]	Selected bus ¹													
00	All channels: counter bus[A]													
01	Channels 0 to 7: counter bus[B]													
	Channels 8 to 15: counter bus[C]													
	Channels 16 to 23: counter bus[D]													
	Channels 24 to 31: counter bus[E]													
10	Reserved													
11	All channels: internal counter													
23 EDSEL	Edge Selection bit. For input modes, the EDSEL bit selects whether the internal counter is triggered by both edges of a pulse or just by a single edge as defined by the EDPOL bit . When not shown in the mode of operation description, this bit has no effect. 0 – Single edge triggering defined by the EDPOL bit 1 – Both edges triggering{mts_ipi_slv_saic} For GPIO in mode, the EDSEL bit selects if a FLAG can be generated. 0 – A FLAG is generated as defined by the EDPOL bit 1 – No FLAG is generated For SAOC mode, the EDSEL bit selects the behavior of the output flip-flop at each match. 0 – The EDPOL value is transferred to the output flip-flop 1 – The output flip-flop is toggled													
24 EDPOL	Edge Polarity bit. For input modes (except QDEC mode), the EDPOL bit asserts which edge triggers either the internal counter or an input capture or a FLAG . When not shown in the mode of operation description, this bit has no effect. 0 – Trigger on a falling edge 1 – Trigger on a rising edge NOTE: UC[n-1] EDPOL bit selects which edge clocks the internal counter of UC[n] NOTE: In order to operate properly, EDPOL bit must contain the same value in UC[n] and UC[n-1] For output modes, the EDPOL bit is used to select the logic level on the output pin . 0 – A match on comparator A clears the output flip-flop, while a match on comparator B sets it													

Table continues on the next page...

eMIOS_Cn field descriptions (continued)

Field	Description																																																										
	1 – A match on comparator A sets the output flip-flop, while a match on comparator B clears it																																																										
25–31 MODE	Mode selection bits. The MODE bits select the mode of operation of the Unified Channel, as shown in the following table. Values not listed are reserved. NOTE: If a reserved value is written to the mode bit field, the results are unpredictable. Table 36-125. UC MODE bits <table> <tr> <th>MODE[0:6]</th><th>Mode of operation¹</th></tr> <tr><td>0000000</td><td>General purpose Input/Output mode (input)</td></tr> <tr><td>0000001</td><td>General purpose Input/Output mode (output)</td></tr> <tr><td>0000010</td><td>Single Action Input Capture</td></tr> <tr><td>0000011</td><td>Single Action Output Compare</td></tr> <tr><td>0000100</td><td>Input Pulse Width Measurement</td></tr> <tr><td>0000101</td><td>Input Period Measurement</td></tr> <tr><td>0000110</td><td>Double Action Output compare (with FLAG set on B match)</td></tr> <tr><td>0000111</td><td>Double Action Output compare (with FLAG set on both match)</td></tr> <tr><td>0001000</td><td>Pulse/Edge Accumulation (continuous)</td></tr> <tr><td>0001001</td><td>Pulse/Edge Accumulation (single shot)</td></tr> <tr><td>0001010</td><td>Pulse/Edge Counting (continuous)</td></tr> <tr><td>0001011</td><td>Pulse/Edge Counting (single shot)</td></tr> <tr><td>0001100</td><td>Quadrature Decode (for count and direction encoders type)</td></tr> <tr><td>0001101</td><td>Quadrature Decode (for phase_A and phase_B encoders type)</td></tr> <tr><td>0001110</td><td>Windowed Programmable Time Accumulation</td></tr> <tr><td>0001111</td><td>Reserved</td></tr> <tr><td>001000b²</td><td>Modulus Counter (Up counter with clear on match start)</td></tr> <tr><td>001001b</td><td>Modulus Counter (Up counter with clear on match end)</td></tr> <tr><td>00101bb</td><td>Modulus Counter (Up/Down counter)</td></tr> <tr><td>00110b0</td><td>Output Pulse Width and Frequency Modulation (immediate update)</td></tr> <tr><td>00110b1</td><td>Output Pulse Width and Frequency Modulation (next period update)</td></tr> <tr><td>00111b0</td><td>Center Aligned Output Pulse Width Modulation (with trail edge dead-time)</td></tr> <tr><td>00111b1</td><td>Center Aligned Output Pulse Width Modulation (with lead edge dead-time)</td></tr> <tr><td>01000b0</td><td>Output Pulse Width Modulation (immediate update)</td></tr> <tr><td>01000b1</td><td>Output Pulse Width Modulation (next period update)</td></tr> <tr><td>0100100 through 0100101</td><td>Reserved</td></tr> <tr><td>0100110</td><td>Output Pulse Width Modulation with Trigger</td></tr> <tr><td>0100111 through 1001111</td><td>Reserved</td></tr> </table>	MODE[0:6]	Mode of operation ¹	0000000	General purpose Input/Output mode (input)	0000001	General purpose Input/Output mode (output)	0000010	Single Action Input Capture	0000011	Single Action Output Compare	0000100	Input Pulse Width Measurement	0000101	Input Period Measurement	0000110	Double Action Output compare (with FLAG set on B match)	0000111	Double Action Output compare (with FLAG set on both match)	0001000	Pulse/Edge Accumulation (continuous)	0001001	Pulse/Edge Accumulation (single shot)	0001010	Pulse/Edge Counting (continuous)	0001011	Pulse/Edge Counting (single shot)	0001100	Quadrature Decode (for count and direction encoders type)	0001101	Quadrature Decode (for phase_A and phase_B encoders type)	0001110	Windowed Programmable Time Accumulation	0001111	Reserved	001000b ²	Modulus Counter (Up counter with clear on match start)	001001b	Modulus Counter (Up counter with clear on match end)	00101bb	Modulus Counter (Up/Down counter)	00110b0	Output Pulse Width and Frequency Modulation (immediate update)	00110b1	Output Pulse Width and Frequency Modulation (next period update)	00111b0	Center Aligned Output Pulse Width Modulation (with trail edge dead-time)	00111b1	Center Aligned Output Pulse Width Modulation (with lead edge dead-time)	01000b0	Output Pulse Width Modulation (immediate update)	01000b1	Output Pulse Width Modulation (next period update)	0100100 through 0100101	Reserved	0100110	Output Pulse Width Modulation with Trigger	0100111 through 1001111	Reserved
MODE[0:6]	Mode of operation ¹																																																										
0000000	General purpose Input/Output mode (input)																																																										
0000001	General purpose Input/Output mode (output)																																																										
0000010	Single Action Input Capture																																																										
0000011	Single Action Output Compare																																																										
0000100	Input Pulse Width Measurement																																																										
0000101	Input Period Measurement																																																										
0000110	Double Action Output compare (with FLAG set on B match)																																																										
0000111	Double Action Output compare (with FLAG set on both match)																																																										
0001000	Pulse/Edge Accumulation (continuous)																																																										
0001001	Pulse/Edge Accumulation (single shot)																																																										
0001010	Pulse/Edge Counting (continuous)																																																										
0001011	Pulse/Edge Counting (single shot)																																																										
0001100	Quadrature Decode (for count and direction encoders type)																																																										
0001101	Quadrature Decode (for phase_A and phase_B encoders type)																																																										
0001110	Windowed Programmable Time Accumulation																																																										
0001111	Reserved																																																										
001000b ²	Modulus Counter (Up counter with clear on match start)																																																										
001001b	Modulus Counter (Up counter with clear on match end)																																																										
00101bb	Modulus Counter (Up/Down counter)																																																										
00110b0	Output Pulse Width and Frequency Modulation (immediate update)																																																										
00110b1	Output Pulse Width and Frequency Modulation (next period update)																																																										
00111b0	Center Aligned Output Pulse Width Modulation (with trail edge dead-time)																																																										
00111b1	Center Aligned Output Pulse Width Modulation (with lead edge dead-time)																																																										
01000b0	Output Pulse Width Modulation (immediate update)																																																										
01000b1	Output Pulse Width Modulation (next period update)																																																										
0100100 through 0100101	Reserved																																																										
0100110	Output Pulse Width Modulation with Trigger																																																										
0100111 through 1001111	Reserved																																																										

Table continues on the next page...

eMIOS_Cn field descriptions (continued)

Field	Description	
	Table 36-125. UC MODE bits (continued)	
	MODE[0:6]	Mode of operation¹
	101000b	Modulus Counter Buffered (Up counter)
	101001b	Reserved
	10101bb	Modulus Counter Buffered (Up/Down counter)
	10110b0	Output Pulse Width and Frequency Modulation Buffered
	10110b1	Reserved
	10111b0	Center Aligned Output Pulse Width Modulation Buffered (with trail edge dead-time)
	10111b1	Center Aligned Output Pulse Width Modulation Buffered (with lead edge dead-time)
	11000b0	Output Pulse Width Modulation Buffered
	1100001 through 1111111	Reserved
	1. Not all modes of operation are supported. See the configuration information for details. 2. b = adjust parameters for the mode of operation. Refer to “UC Modes of Operation” section for details.	

1. Filter latency (the difference in time between the input and the response) is 3 clock edges.
2. The input signal is synchronized before arriving to the digital filter.
3. Not all devices support all counter buses. See the Device Configuration chapter for details.
4. Not all modes of operation are supported. See the configuration information for details.
5. b = adjust parameters for the mode of operation. Refer to “UC Modes of Operation” section for details.

36.5.11 eMIOS UC Status register n (eMIOS_Sn)

This section describes the register for channels 16-23. For channels 0-7, see [eMIOS UC Status register n \(eMIOS_Sn\)](#).

The UC Status register EMIOS_Sn gathers bits reflecting the status of the UC input/output signals and the overflow condition of the internal counter.

Address: 0h base + 230h offset + (32d × i), where i=0d to 7d

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	OVR	0														
W	w1c															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	OVFL	0												UCIN	UCOUT	FLAG
W	w1c															w1c
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	*	0	0

* Notes:

- UCIN field: Reset value is undefined.

eMIOS_*Sn* field descriptions

Field	Description
0 OVR	Overflow bit. The OVR bit indicates that FLAG generation occurred when the FLAG bit was already set . The OVR bit can be cleared either by clearing the FLAG bit or by writing a 1 to the OVR bit. 0 Overflow has not occurred 1 Overflow has occurred
1–15 Reserved	Reserved. This field is reserved. This read-only field is reserved and always has the value 0.
16 OVFL	Overflow bit. The OVFL bit indicates that an overflow has occurred in the internal counter. The OVFL bit must be cleared by software writing a 1 to it. 0 No overflow 1 An overflow had occurred
17–28 Reserved	Reserved. This field is reserved. This read-only field is reserved and always has the value 0.
29 UCIN	Unified Channel Input pin bit. The UCIN bit reflects the input pin state after being filtered and synchronized.
30 UCOUT	Unified Channel Output pin bit. The UCOUT bit reflects the output pin state .
31 FLAG	FLAG bit. The FLAG bit is set when an input capture or a match event in the comparators occurred. The FLAG bit must be cleared by writing a 1 to it.

Table continues on the next page...

eMIOS_Sn field descriptions (continued)

Field	Description
0	FLAG cleared
1	FLAG set event has occurred

36.5.12 eMIOS Alternate Address register n (eMIOS_ALTAn)

This section describes the register for channels 16-23. In all other respects, this register is identical to the register for channels 0-7. See that register's description: [eMIOS Alternate Address register n \(eMIOS_ALTAn\)](#).

Address: 0h base + 234h offset + (32d × i), where i=0d to 7d

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0								ALTA																							
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	

eMIOS_ALTAn field descriptions

Field	Description
0–7 Reserved	Reserved. This field is reserved. This read-only field is reserved and always has the value 0.
8–31 ALTA	See text above register figure.

36.6 Functional Description

The eMIOS provides independent channels (Unified) that can be configured and accessed by a host MCU. Up to five time bases can be shared by the channels.

The eMIOS module is based on a multi-bus timer architecture in which several timer channels are used to drive counter buses that are shared among the channels. Up to five timebases can be shared by the channels through five counter buses (counter bus A or F, B, C, D and E) and each channel can generate its own timebase. There are five counter buses in the module: one global counter bus, shared by all channels and selectable among time bases A or F, and four local counter buses (B, C, D, and E), each one dedicated to a slice of 8 channels. Counter bus A is referred to as the global counter buse. Counter buses B, C, D, and E are the local counter buses.

The eMIOS counter buses are driven by channels in specific locations. Global counter bus A is driven by the channel in channel slot [23]. Counter buses B, C, D, and E are driven by channels in slots 0, 8, 16, and 24, respectively. Counter bus A or F drives all channels. Counter bus B drives channels in slots from 0 through 7. Counter bus C drives channels in slots from 8 through 15. Counter bus D drives channels in slots from 16 through 23. Counter bus E drives channels from 24 through 31.

The eMIOS block is reset asynchronously. All registers are cleared on reset.

The following figure shows an eMIOS block configured with a mix of Unified Channels. In this configuration example, each one of the channels can be driven by the respective local bus—B for the [7:0] slice and D for the [23:16] slice—or by the global counter bus A. Note that there are Unified Channels with no external pin associated. These channels are used to trigger internal SoC signals such as interrupt requests or DMA transfer requests. Since the channel flags are also connected to the block top-level pins, they can be used as trigger signals for other blocks at chip level.

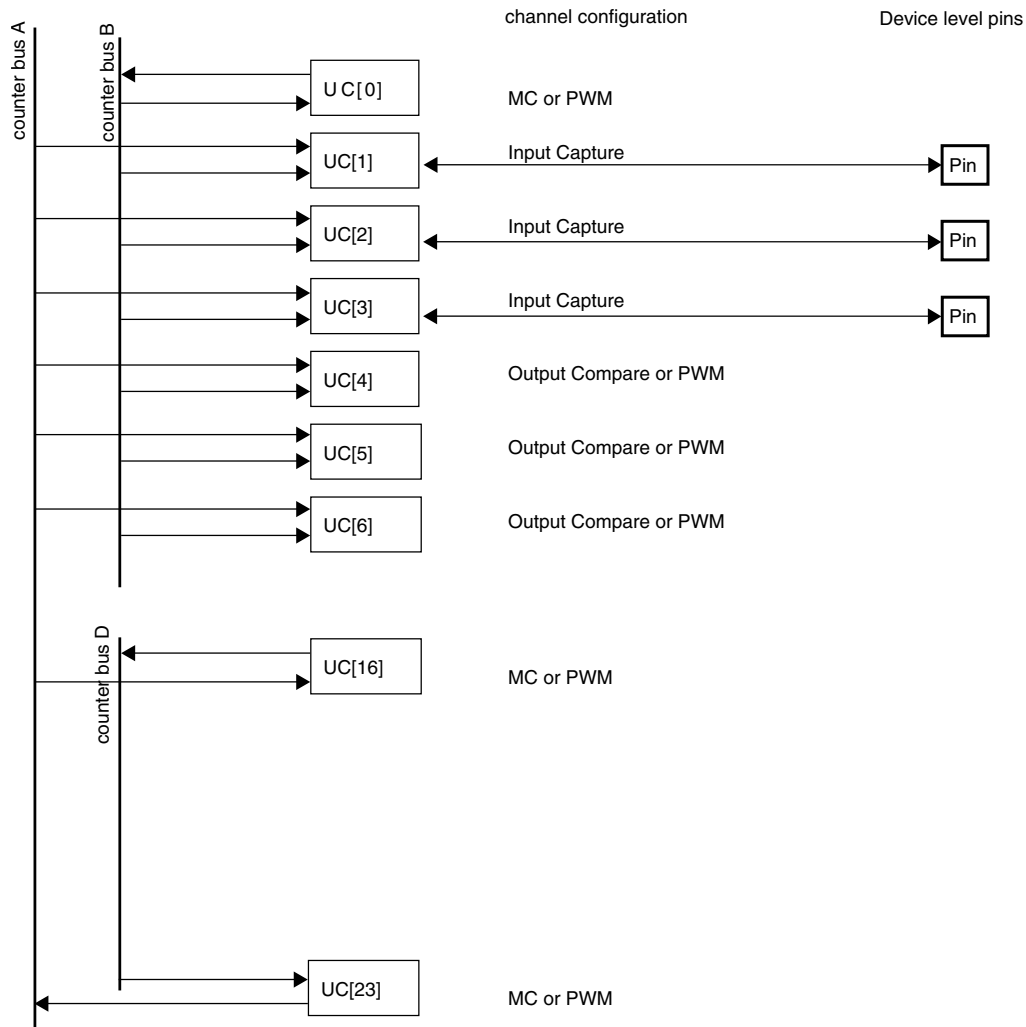


Figure 36-113. eMIOS example configuration using Unified Channels

36.6.1 Unified Channel (UC)

Figure 36-114 shows the Unified Channel block diagram. Each Unified Channel consists of the following:

- A counter bus selector, which selects the timebase to be used by the channel for all timing functions
- A programmable clock prescaler
- Two double-buffered data registers, A and B, that allow up to two input capture and/or output compare events to occur before software intervention is needed
- Two comparators (equal only), A and B, that compare the selected counter bus with the value in the data registers
- An internal counter, which can be used as a local time base or to count input events
- Programmable input filter, which ensures that only valid pin transitions are received by channel
- Programmable input edge detector, which detects the rising, falling, or either edge
- An output flip-flop, which holds the logic level to be applied to the output pin
- eMIOS Status and Control register
- An Output Disable Input selector, which selects the Output Disable Input signal that will be used as output disable

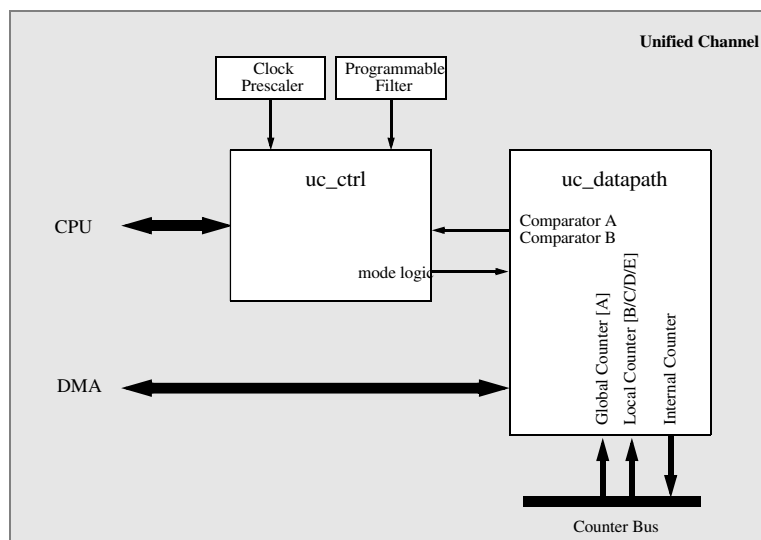


Figure 36-114. Unified Channel block diagram

Figure 36-115 shows both the Unified Channel control and datapath block diagrams. The control sub-block is responsible for the generation of signals to control the multiplexers in the datapath sub-block. Each mode is implemented by a dedicated logic independent from others modes, thus allowing to optimize the logic by disabling the mode and therefore its associated logic. Targeting the logic optimization, a set of registers is shared by the modes thus providing sequential events to be stored.

The datapath block provides the channel A and B registers, the internal timebase and comparators. Multiplexers select the input of comparators and data for the registers inputs, thus configuring the datapath to implement the channel modes. The outputs of the A and B comparators are connected to the uc_ctrl control block.

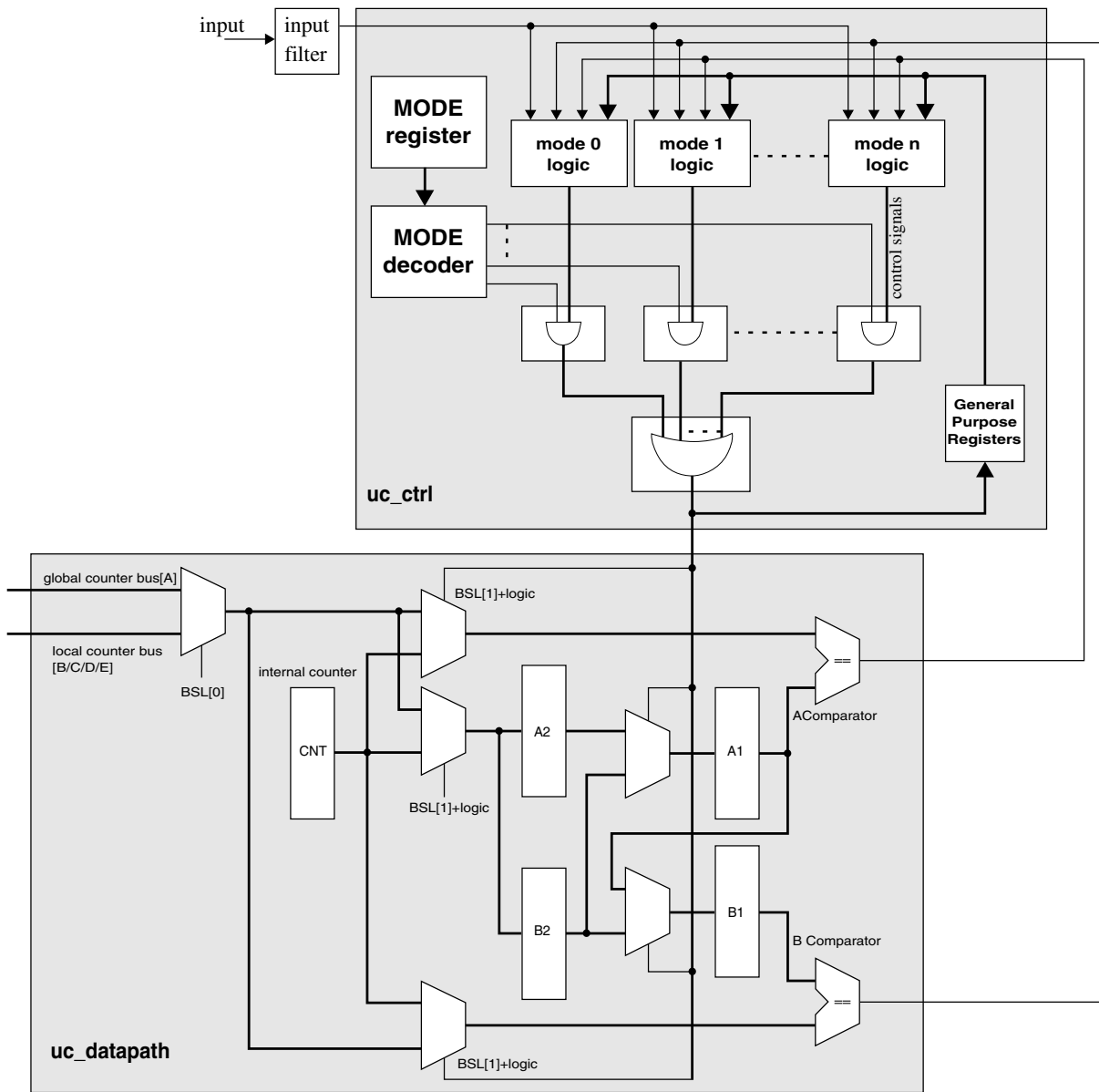


Figure 36-115. Unified Channel control and datapath block diagrams

36.6.1.1 Unified Channel modes of operation

The mode of operation of the Unified Channel is determined by the mode select bits MODE[0:6] in the EMIOS_C_n register.

When entering an output mode (except for GPIO mode), the output flip-flop is set to the disabled state according to EMIOSCn[ODIS].

As the internal counter, EMIOSCNTn, runs in all modes (except for GPIO mode) and can be used as a timebase if the resource is not used in the current mode.

In order to provide smooth waveform generation even when A and B registers are immediately changed, it is available with the MCB, OPWFMB, OPWMB and OPWFMB modes. In these modes, A and B registers are double buffered. They are presented in separate sections because there are several basic differences between these and each corresponding nonbuffered mode.

36.6.1.1.1 General-Purpose Input/Output (GPIO) mode

In GPIO mode, all input capture and output compare functions of the Unified Channel are disabled, and the internal counter (EMIOSCNTn register) is cleared and disabled. All control bits remain accessible. In order to prepare the UC for a new operation mode, writing to registers EMIOSAn or EMIOSBn stores the same value in registers A1/A2 or B1/B2, respectively. Writing to the EMIOSALTAn register stores a value only in register A2.

MODE[6] bit selects between input (MODE[6] = 0) and output (MODE[6] = 1) modes.

When changing MODE[0:6], the application software must first enter GPIO mode to reset the Unified Channel's internal functions properly. Failure to do this could lead to invalid and unexpected output compare or input capture results or the FLAGS being set incorrectly.

In GPIO input mode (MODE[0:6] = 0000000), FLAG generation is determined according to EDPOL and EDSEL bits and the input pin status can be determined by reading the UCIN bit.

In GPIO output mode (MODE[0:6] = 0000001), the Unified Channel is used as a single output port pin and the value of the EDPOL bit is permanently transferred to the output flip-flop.

36.6.1.1.2 Single Action Input Capture (SAIC) Mode

In SAIC mode (MODE[0:6]=0000010), when a triggering event occurs on the input pin, the value on the selected time base is captured into register A2. The FLAG bit is set along with the capture event to indicate that an input capture has occurred. Register EMIOSA[n] returns the value of register A2. As soon as the SAIC mode is entered coming out from GPIO mode the channel is ready to capture events. The events are captured as soon as they occur thus reading register A always returns the value of the latest captured event. Subsequent captures are enabled with no need of further reads from EMIOSA[n] register. The FLAG is set at any time a new event is captured.

The input capture is triggered by a rising, falling or either edges in the input pin, as configured by EDPOL and EDSEL bits in EMIOSC[n] register.

Figure 36-116 and Figure 36-117 show how the Unified Channel can be used for input capture.

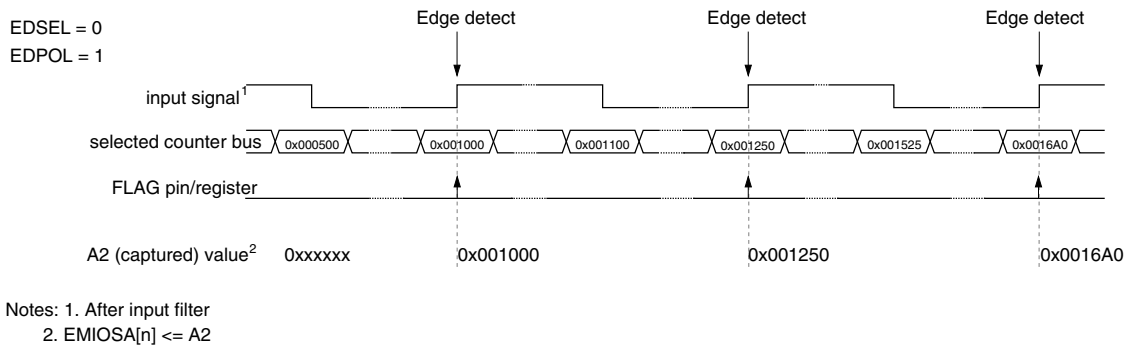


Figure 36-116. Single Action Input Capture with rising edge triggering example

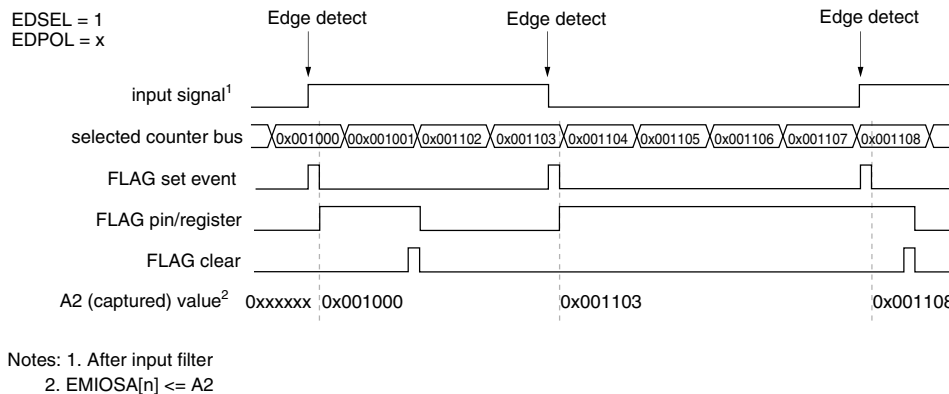


Figure 36-117. Single Action Input Capture with both edges triggering example

36.6.1.1.3 Single Action Output Compare (SAOC) Mode

In SAOC mode (MODE[0:6]=0000011) a match value is loaded in register A2 and then immediately transferred to register A1 to be compared with the selected time base. When a match occurs, the EDSEL bit selects whether the output flip-flop is toggled or the value

in EDPOL is transferred to it. Along with the match the FLAG bit is set to indicate that the output compare match has occurred. Writing to register EMIOSA[n] stores the value in register A2 and reading to register EMIOSA[n] returns the value of register A1.

An output compare match can be simulated in software by setting the FORCMA bit in EMIOSC[n] register. In this case, the FLAG bit is not set.

When SAOC mode is entered coming out from GPIO mode, the output flip-flop is set to the complement of the EDPOL bit in the EMIOSC[n] register.

Counter bus can be either internal or external and is selected through BSL[0:1] bits.

Figure 36-118 and Figure 36-119 show how the Unified Channel can be used to perform a single output compare with EDPOL value being transferred to the output flip-flop and toggling the output flip-flop at each match, respectively. Note that once in SAOC mode the matches are enabled thus the desired match value on register A1 must be written before the mode is entered. A1 register can be updated at any time thus modifying the match value which will reflect in the output signal generated by the channel. Subsequent matches are enabled with no need of further writes to EMIOSA[n] register. The FLAG is set at the same time a match occurs (see Figure 36-120).

Note

The channel internal counter in SAOC mode is free-running. It starts counting as soon as the SAOC mode is entered.

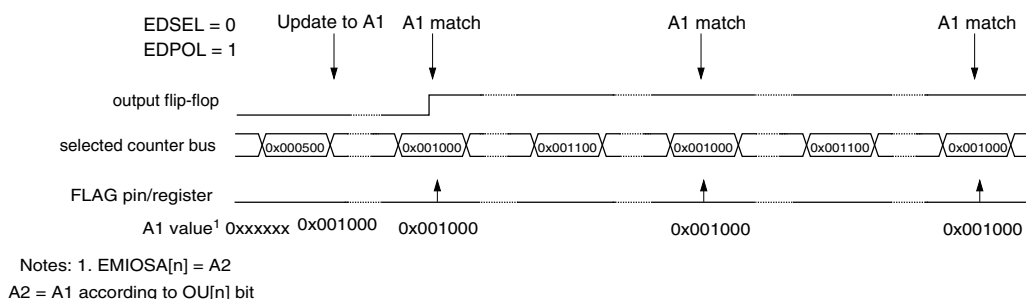


Figure 36-118. SAOC example with EDPOL value being transferred to the output flip-flop

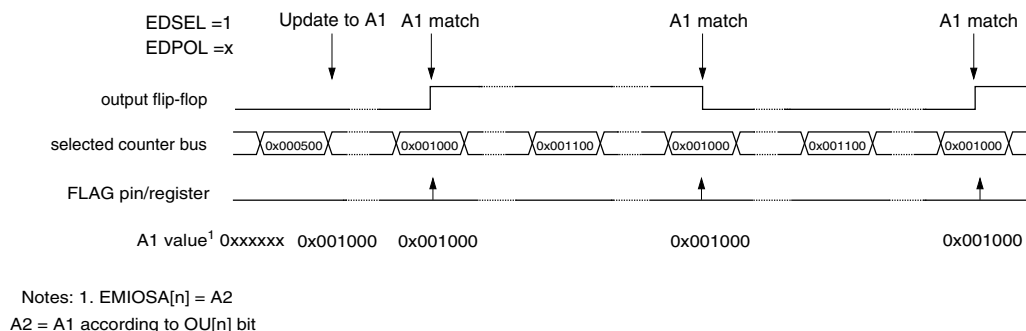


Figure 36-119. SAOC example toggling the output flip-flop

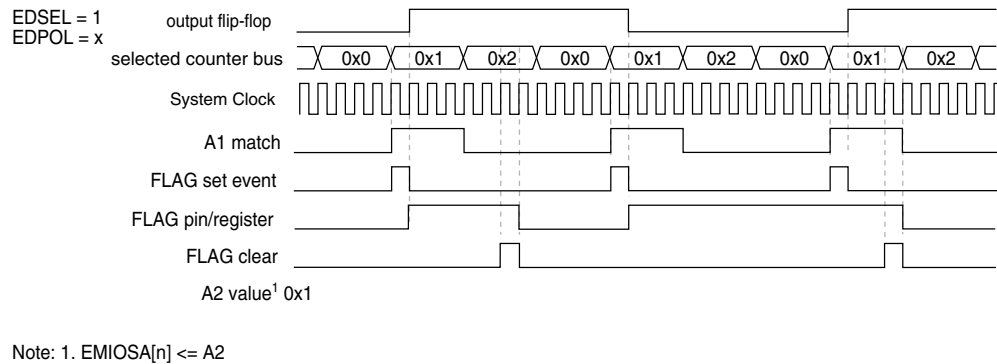


Figure 36-120. SAOC example with flag behavior

36.6.1.1.4 Input Pulse Width Measurement (IPWM) Mode

The IPWM mode (MODE[0:6]=0000100) allows the measurement of the width of a positive or negative pulse by capturing the leading edge on register B1 and the trailing edge on register A2. Successive captures are done on consecutive edges of opposite polarity. The leading edge sensitivity (i.e., pulse polarity) is selected by EDPOL bit in the EMIOSC[n] register. Registers EMIOSA[n] and EMIOSB[n] return the values in register A2 and B1, respectively.

The capture function of register A2 remains disabled until the first leading edge triggers the first input capture on register B2. When this leading edge is detected, the count value of the selected time base is latched into register B2; the FLAG bit is not set. When the trailing edge is detected, the count value of the selected time base is latched into register A2 and, at the same time, the FLAG bit is set and the content of register B2 is transferred to register B1 and to register A1.

If subsequent input capture events occur while the corresponding FLAG bit is set, registers A2, B1 and A1 will be updated with the latest captured values and the FLAG will remain set. Registers EMIOSA[n] and EMIOSB[n] return the value in registers A2 and B1, respectively.

In order to guarantee coherent access, reading EMIOSA[n] forces B1 be updated with the content of register A1. At the same time transfers between B2 and B1 are disabled until the next read of EMIOSB[n] register. Reading EMIOSB[n] register forces B1 be updated with A1 register content and re-enables transfers from B2 to B1, to take effect at the next trailing edge capture. Transfers from B2 to A1 are not blocked at any time.

The input pulse width is calculated by subtracting the value in B1 from A2.

The following figure shows how the Unified Channel can be used for input pulse width measurement.

Downloaded from <http://www.sagepub.com> at NANYANG TECH UNIV LIBRARY on June 11, 2015

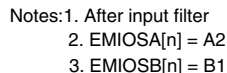


Figure 36-121. Input Pulse Width Measurement example

The following figure shows the A1 and B1 updates when EMIOSA[n] and EMIO SB[n] register reads occur. Note that A1 register has always coherent data related to A2 register. Note also that when EMIOSA[n] read is performed B1 register is loaded with A1 register content. This guarantee that the data in register B1 has always the coherent data related to the last EMIOSA[n] read. The B1 register updates remains locked until EMIO SB[n] read occurs. If EMIOSA[n] read is performed B1 is updated with A1 register content even if B1 update is locked by a previous EMIOSA[n] read operation.

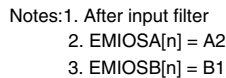


Figure 36-122. B1 and A1 updates at EMIOSA[n] and EMIO SB[n] reads

Reading EMIOS_A_n followed by EMIOS_B_n always provides coherent data. If coherent data is not required for any reason, the sequence of reads should be inverted, therefore EMIOSB[n] should be read prior to EMIOSA[n] register. Note that even in this case B1 register updates will be blocked after EMIOSA[n] read, thus a second EMIOSB[n] is required in order to release B1 register updates.

36.6.1.1.5 Input Period Measurement (IPM) Mode

The IPM mode (MODE[0:6]=0000101) allows the measurement of the period of an input signal by capturing two consecutive rising edges or two consecutive falling edges.

Successive input captures are done on consecutive edges of the same polarity. The edge polarity is defined by the EDPOL bit in the EMIOSC[n] register.

Note

The input signal must have at least four system clock cycles period in order to be properly captured by the synchronization logic at the channel input even if the input filter is in by-pass mode.

When the first edge of selected polarity is detected, the selected time base is latched into the registers A2 and B2, and the data previously held in register B2 is transferred to register B1. On this first capture the FLAG line is not set, and the values in registers B1 is meaningless. On the second and subsequent captures, the FLAG line is set and data in register B2 is transferred to register B1.

When the second edge of the same polarity is detected, the counter bus value is latched into registers A2 and B2, the data previously held in register B2 is transferred to data register B1 and to register A1. The FLAG bit is set to indicate the start and end points of a complete period have been captured. This sequence of events is repeated for each subsequent capture. Registers EMIOSA[n] and EMIOSB[n] return the values in register A2 and B1, respectively.

In order to allow coherent data, reading EMIOSA[n] forces A1 content be transferred to B1 register and disables transfers between B2 and B1. These transfers are disabled until the next read of the EMIOSB[n] register. Reading EMIOSB[n] register forces A1 content to be transferred to B1 and re-enables transfers from B2 to B1, to take effect at the next edge capture.

The input pulse period is calculated by subtracting the value in B1 from A2.

The following figure shows how the Unified Channel can be used for input period measurement.

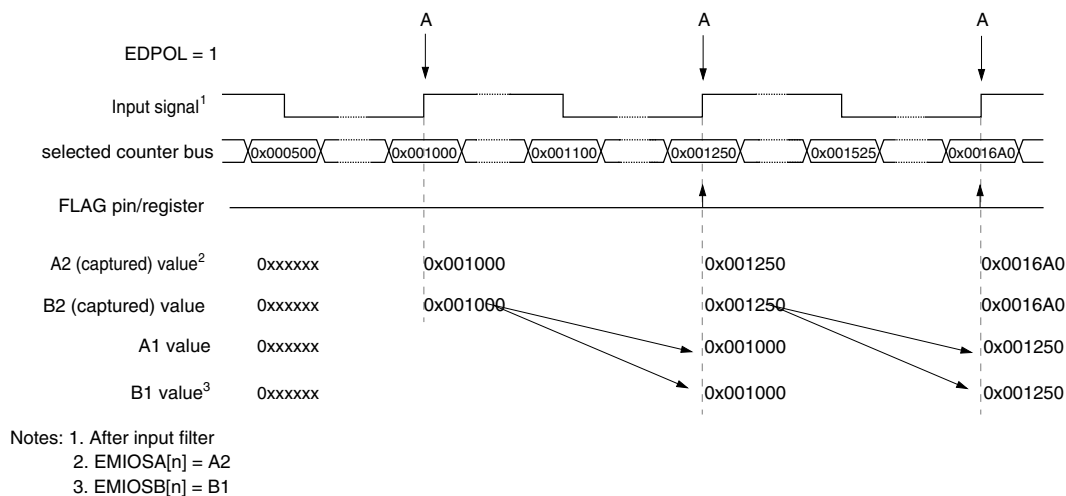


Figure 36-123. Input Period Measurement example

Figure 36-124 describes the A1 and B1 register updates when EMIOSA[n] and EMIOSB[n] read operations are performed. When EMIOSA[n] read occurs the content of A1 is transferred to B1 thus providing coherent data in A2 and B1 registers. Transfers from B2 to B1 are then blocked until EMIOSB[n] is read. After EMIOSB[n] is read, register A1 content is transferred to register B1 and the transfers from B2 to B1 are re-enabled to occur at the transfer edges, which is the leading edge in the following example figure.

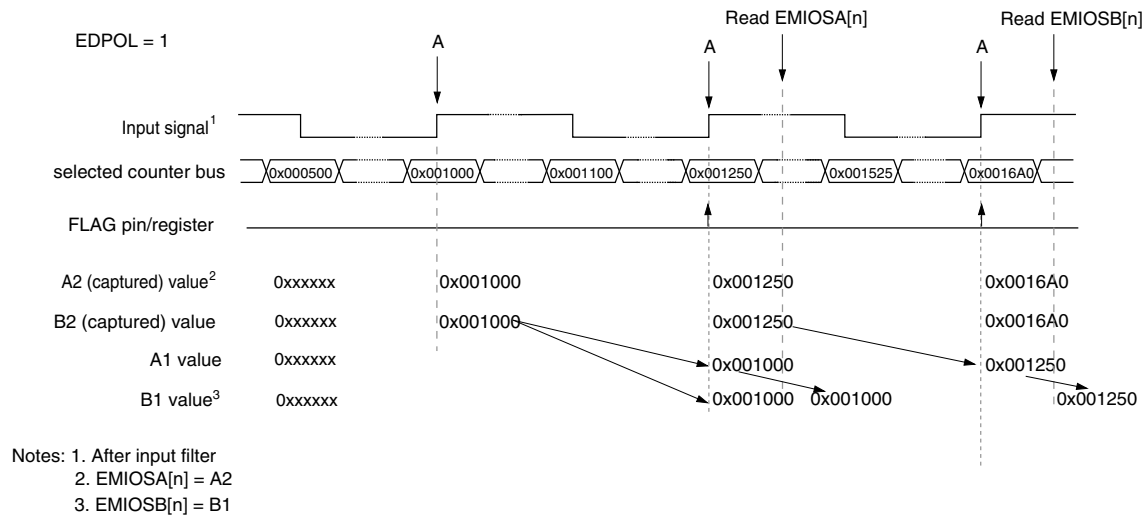


Figure 36-124. A1 and B1 updates at EMIOSA[n] and EMIOSB[n] reads

36.6.1.1.6 Double Action Output Compare (DAOC) Mode

In the DAOC mode, the leading and trailing edges of the variable pulse width output are generated by matches occurring on comparators A and B. There is no restriction concerning the order in which A and B matches occur.

When the DAOC mode is entered, coming out from GPIO mode both comparators are disabled and the output flip-flop is set to the complement of the EDPOL bit in the EMIOSC[n] register.

Data written to A2 and B2 are transferred to A1 and B1, respectively, on the next system clock cycle if OU[n] bit of EMIOSOUDIS register is cleared (see [Figure 36-151](#)). The transfer is blocked if OU[n] bit is set. Comparator A is enabled only after the transfer to A1 register occurs and is disabled on the next A match. Comparator B is enabled only after the transfer to B1 register occurs and is disabled on the next B match. Comparators A and B are enabled and disabled independently.

The output flip-flop is set to the value of EDPOL when a match occurs on comparator A and to the complement of EDPOL when a match occurs on comparator B.

MODE[6] controls if the FLAG is set on both matches (MODE[0:6]=0000111) or just on the B match (MODE[0:6]=0000110). FLAG bit assertion depends on comparator enabling.

If subsequent enabled output compares occur on registers A1 and B1, pulses will continue to be generated, regardless of the state of the FLAG bit.

At any time, the FORCMA and FORCMB bits allow the software to force the output flip-flop to the level corresponding to a comparison event in comparator A or B, respectively. Note that the FLAG bit is not affected by these forced operations.

Note

If both registers (A1 and B1) are loaded with the same value, the B match prevails concerning the output pin state (output flip-flop is set to the complement of EDPOL), the FLAG bit is set and both comparators are disabled.

The following two tables show how the Unified Channel can be used to generate a single output pulse with FLAG bit being set on the second match or on both matches, respectively.

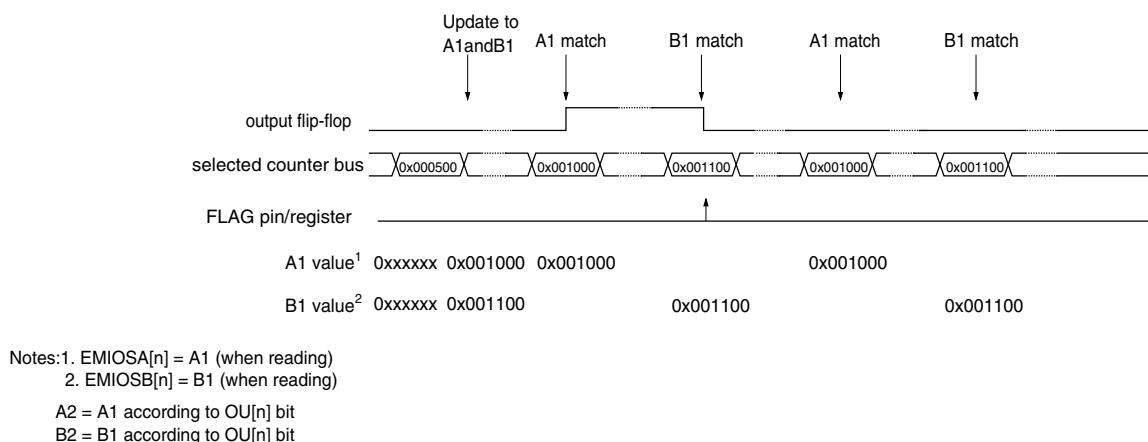


Figure 36-125. Double Action Output Compare with FLAG set on the second match, MODE[0] bit is 0

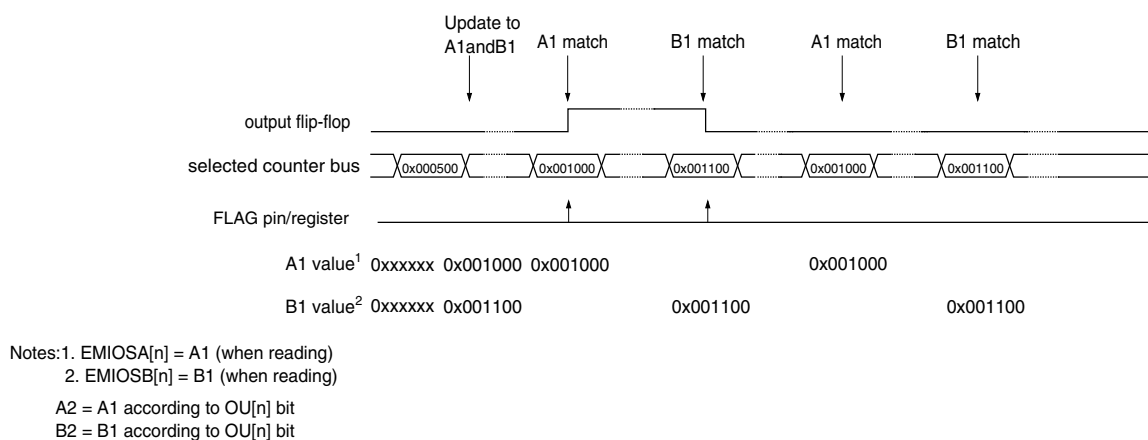


Figure 36-126. Double Action Output Compare with FLAG set on both matches, MODE[0] bit is 1

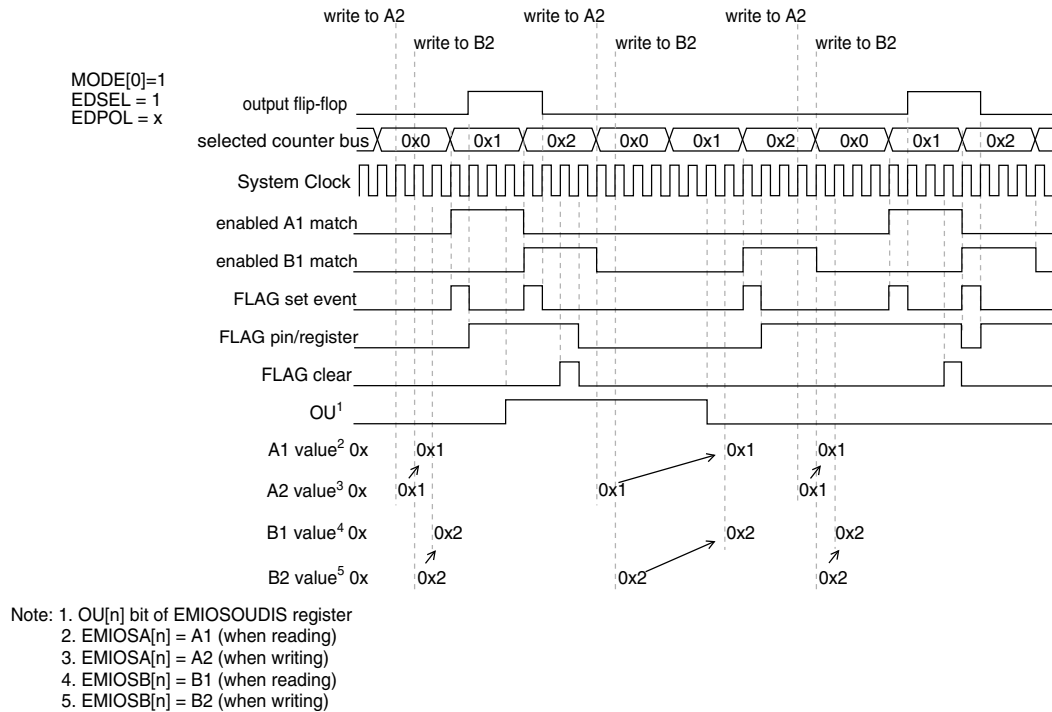


Figure 36-127. DAOC with transfer disabling example

36.6.1.1.7 Pulse/Edge Accumulation (PEA) Mode

The PEA mode returns the time taken to detect a desired number of input events. MODE[6] bit selects between continuous or single shot operation.

After writing to register A1, the internal counter is cleared on the first input event, ready to start counting input events and the selected timebase is latched into register B2. On the match between the internal counter and register A1, a counter bus capture is triggered to register A2 and B2. The data previously held in register B2 is transferred to register B1 and the FLAG bit is set to indicate that an event has occurred. The desired time interval can be determined by subtracting register B1 from A2. Registers EMIOSA[n] and EMIOSB[n] return the values in register A2 and B1, respectively.

As part of the coherency mechanism, reading EMIOSA[n] disables transfers from B2 to B1. These transfers are disabled until the next read of the EMIOS_B_n register. Reading the EMIOSB[n] register re-enables transfers from B2 to B1, to take effect at the next transfer event, as described above. ³

In order to have coherent data in continuous mode of operation the following steps should be performed, assuming FLAG is initially cleared:

3. If B1 was not updated due to B2 to B1 transfer being disabled after reading register EMIOS_A_n, further EMIOS_A_n] and EMIOS_B_n reads will not return coherent data until a new bus capture is triggered to registers A2 and B2. This capture event is indicated by the channel FLAG being asserted. If enabled, the FLAG also generates an interrupt.

Functional Description

1. Wait for FLAG assertion;
2. Read EMIOSA[n] register;
3. Read EMIOSB[n] register;
4. Clear FLAG bit;
5. Return to step 1.

Accumulation cycles may be lost if the read is not performed in a timely manner. Whenever Overrun bit is asserted it means that one or more cycles have been lost.

Triggering of the counter clock (input event) is done by a rising or falling edge or both edges on the input pin. The polarity of the triggering edge is selected by the EDSEL and EDPOL bits in EMIOS_C_n register.

For continuous operation mode (MODE[6] cleared, MODE[0:6]=0001000), the counter is cleared on the next input event after a FLAG generation and continues to operate as described above.

For single shot operation (MODE[6] set, MODE[0:6]=0001001), the counter is not cleared or incremented after a FLAG generation, until a new writing operation to register A is performed.

Figure 36-128 and Figure 36-129 show how the Unified Channel can be used for continuous and single shot pulse/edge accumulation mode.

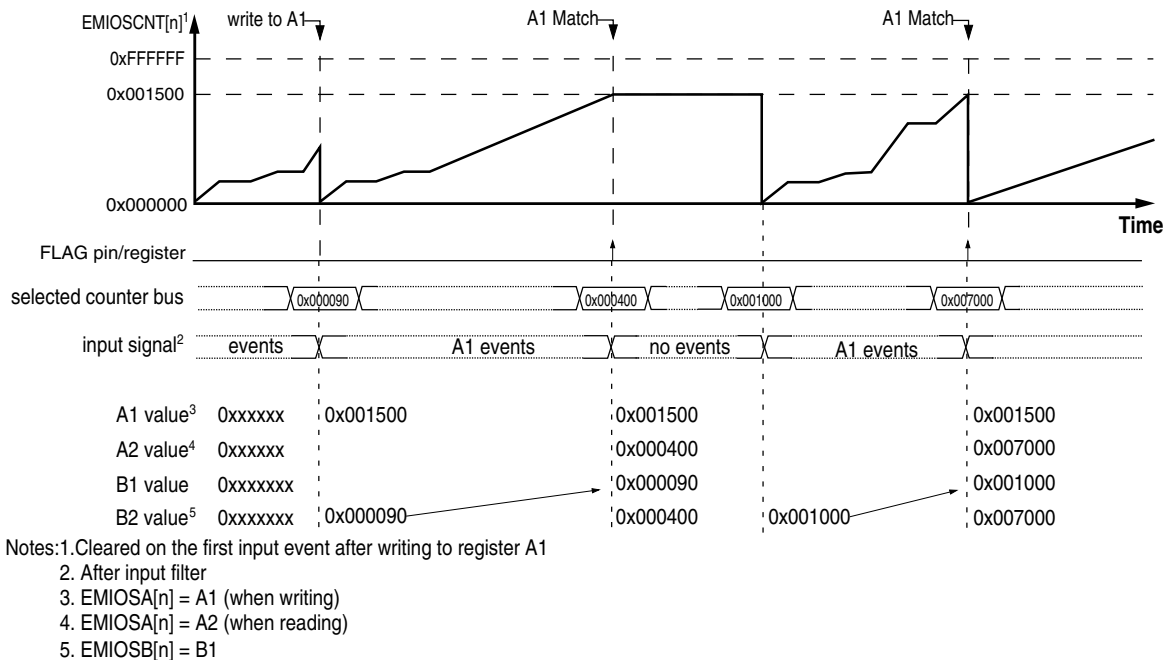


Figure 36-128. Pulse/Edge Accumulation continuous mode example (MODE[6] bit is 0b)

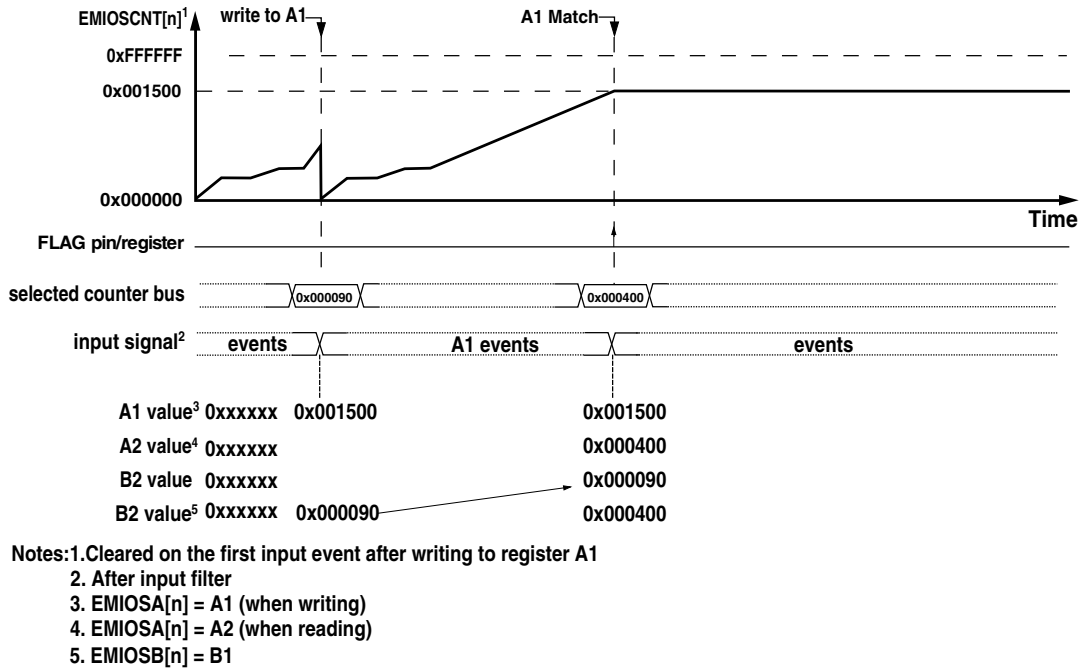


Figure 36-129. Pulse/Edge Accumulation single-shot mode example (MODE[6] bit is 1b)

36.6.1.1.8 Pulse/Edge Counting (PEC) Mode

The PEC mode returns the amount of pulses or edges detected on the input for a desired time window. MODE[6] bit selects between continuous or single shot operation.

Triggering of the internal counter is done by a rising or falling edge or both edges on the input signal. The polarity and the triggering edge is selected by EDSEL and EDPOL bits in EMIOSC[n] register.

Register A1 holds the start time and register B1 holds the stop time for the time window. After writing to register A1, when a match occur between comparator A and the selected timebase, the internal counter is cleared and it is ready to start counting input events. When the time base matches comparator B, the internal counter is disabled and its content is transferred to register A2. At the same time the FLAG bit is set. Reading registers EMIOSCNT[n] or A2 returns the amount of detected pulses.

For continuous operation (MODE[6] cleared, MODE[0:6]=0001010), the next match between comparator A and the selected time base clears the internal counter and counting is enabled again. In order to guarantee coherent measurements when reading EMIOSCNT[n] after the FLAG is set, the software must check if the time base value is out of the time interval defined by registers A1 and B1. Alternatively register A2 always holds the latest available measurement providing coherent data at any time after the first FLAG had occurred. This register is addressed by the alternate address EMIOSALTA[n].

Functional Description

For single shot operation (MODE[6] set, MODE[0:6]=0001011), the next match between comparator A and the selected time base has no effect, until a new write to register A is performed. The EMIOSENT content is also transferred to register A2 when a match in the B comparator occurs.

Figure 36-130 and Figure 36-131 show how the Unified Channel can be used for continuous or single shot pulse/edge counting mode.

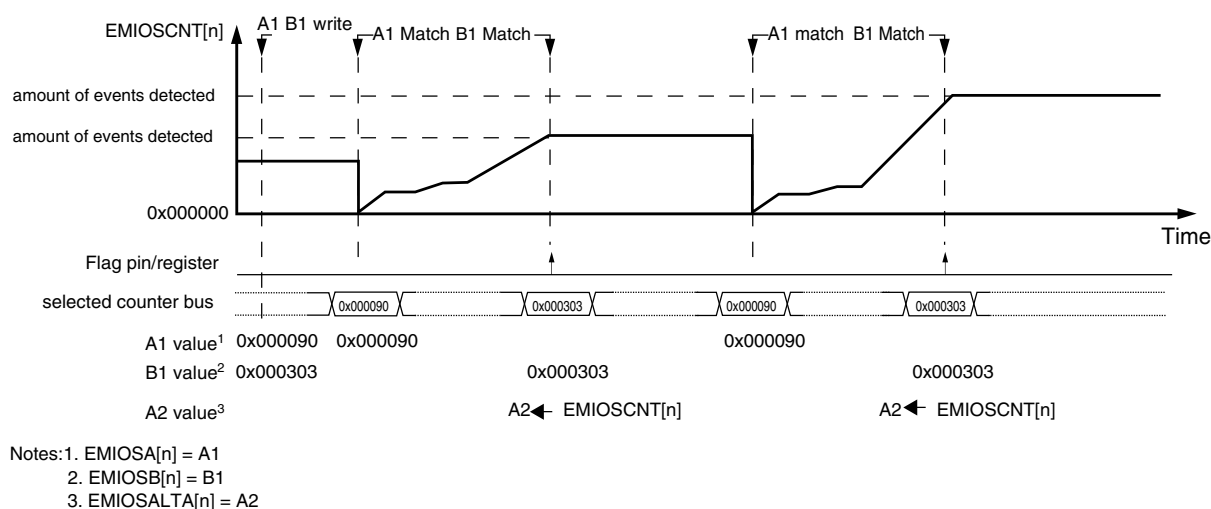


Figure 36-130. Pulse/Edge Counting continuous mode example

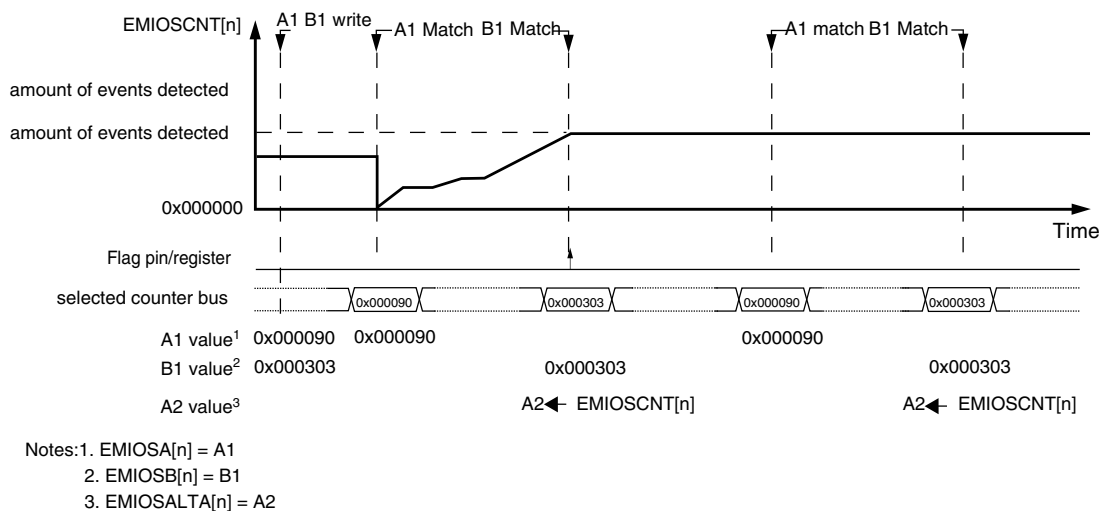


Figure 36-131. Pulse/Edge Counting single-shot mode example

36.6.1.1.9 Quadrature Decode (QDEC) Mode

Quadrature decode mode uses UC[n] operating in QDEC mode and the input programmable filter (IPF) from UC[n-1]. Note that UC[n-1] can be configured, at the same time, to an operation mode that does not use I/O pins, such as MC mode (modulus counter). The connection among the UCs is circular, i.e., when UC[0] is running in QDEC mode, the input programmable filter from UC[23] is being used, for the case of a 24-channel configuration.

This mode generates a FLAG every time the internal counter matches A1 register. The internal counter is automatically selected and is not cleared when entering this mode.

MODE[6] bit selects which type of encoder will be used: *count & direction* encoder or *phase_A and phase_B* encoders.

When operating with *count and direction* encoder (MODE[6] cleared), UC[n] input pin must be connected to the *direction* signal and UC[n-1] input pin must be connected to the *count* signal of the quadrature encoder. UC[n] EDPOL bit selects count direction according to *direction* signal and UC[n-1] EDPOL bit selects if the internal counter is clocked by the rising or falling edge of the *count* signal.

When operating with *phase_A and phase_B* encoder (MODE[6] set), UC[n] input pin must be connected to the *phase_A* signal and UC[n-1] input pin must be connected to the *phase_B* signal of the quadrature encoder. EDPOL bit selects the count direction according to the phase difference between *phase_A* and *phase_B* signals.

Figure 36-132 and Figure 36-133 show two Unified Channels configured to quadrature decode mode for *count & direction* encoder and *phase_A and phase_B* encoders, respectively.

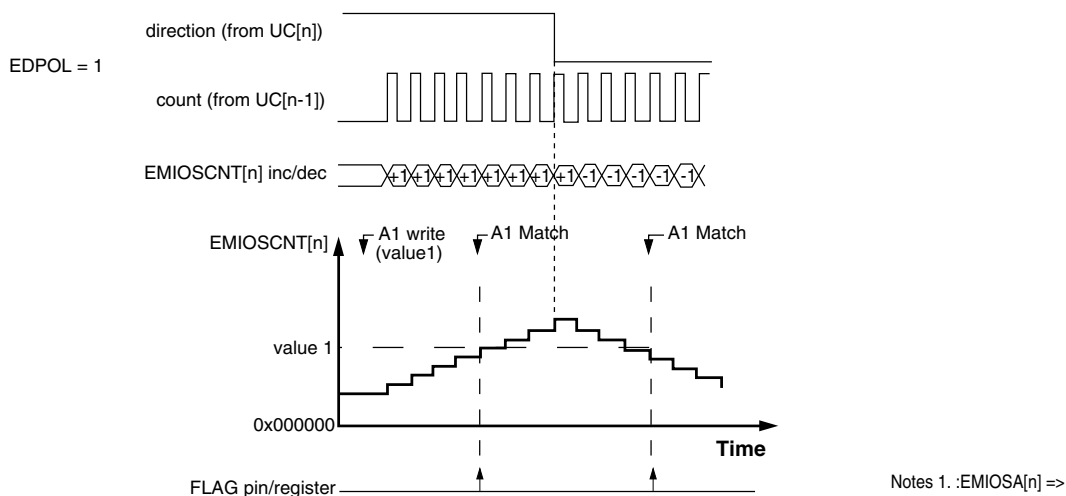


Figure 36-132. Quadrature Decode mode example with *count and direction* encoder

Functional Description

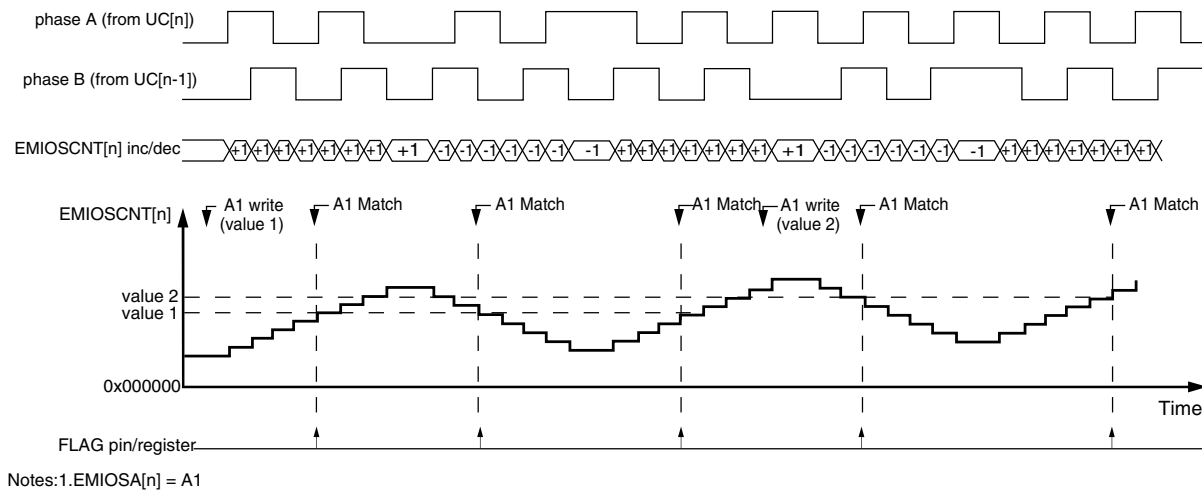


Figure 36-133. Quadrature Decode mode example with *phase_A* and *phase_B* encoder

36.6.1.1.10 Windowed Programmable Time Accumulation (WPTA) Mode

The WPTA mode (MODE[0:6]=0001110) accumulates the sum of the total high time or low time of an input signal over a programmable interval (time window).

The prescaler bits UCPRE[0:1] in EMIOSC[n] register define the increment rate of the internal counter.

Register A1 holds the start time and register B1 holds the stop time of the programmable time interval. When a match occurs between register A and the selected timebase, the internal counter is cleared and it is ready to start counting. The internal counter is used as a time accumulator, i.e., it counts up when the input signal has the same polarity of EDPOL bit in EMIOSC[n] register and does not count otherwise. When a match occurs in comparator B, the internal counter is disabled regardless of the input signal polarity and the FLAG bit is set. At the same time the content of EMIOSCNT[n] is transferred to register A2. Reading registers EMIOSCNT[n] or A2 returns the high or low time of the input signal.

Note that EMIOSCNT[n] is stable only outside the time window defined from A1 to B1 matches, otherwise its content reflects a count in progress and not the final value. Alternatively to EMIOSCNT register A2 returns the latest available measurement. Since this register is updated only at comparator B matches it always contains stable and up-to-date data. In this mode this register is accessible through the alternate register address EMIOSALTA[n].

The following figure shows how the Unified Channel can be used to accumulate high time.

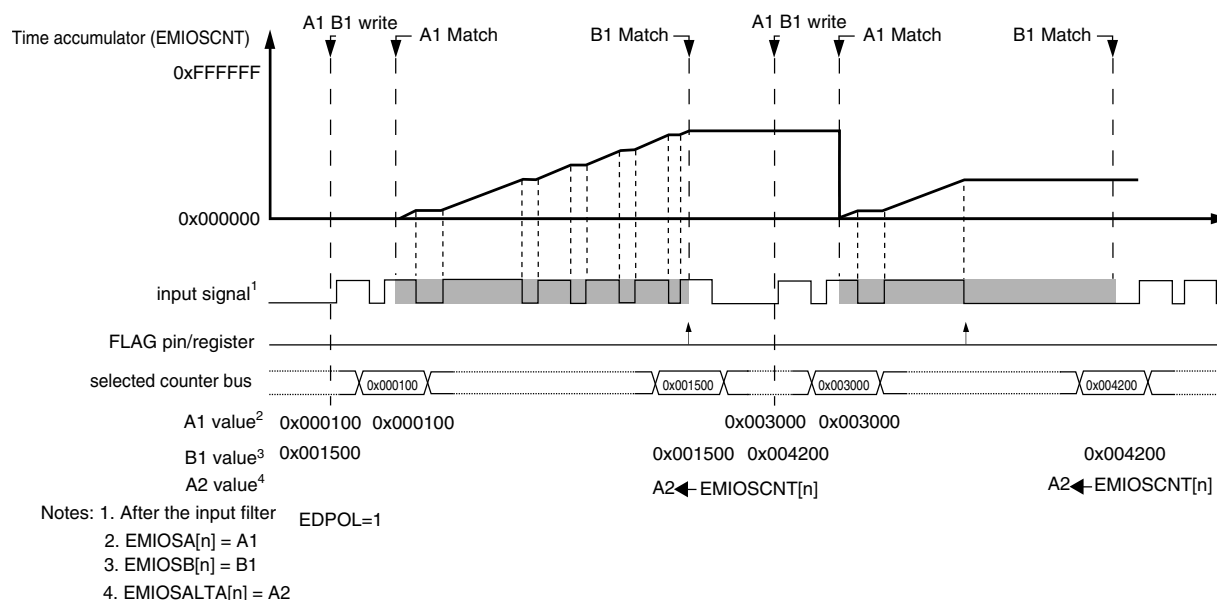


Figure 36-134. Windowed Programmable Time Accumulation example

36.6.1.1.11 Modulus Counter (MC) Mode

The MC mode can be used to provide a time base for a counter bus or as a general purpose timer.

MODE[6] bit selects internal or external clock source when cleared or set, respectively. When external clock is selected, the input signal pin is used as the source and the triggering polarity edge is selected by the EDPOL and EDSEL in the EMIOSC[n] register.

The internal counter counts up from the current value until it matches the value in register A1. Register B1 is cleared and is not accessible to the MCU. MODE[4] bit selects up mode or up/down mode, when cleared or set, respectively.

When in up count mode, a match between the internal counter and register A1 sets the FLAG and clears the internal counter. The timing of those events varies according to the MC mode setup as follows:

- Internal counter clearing on match start (MODE[0:6]=001000b)

- External clock is selected if MODE[0] is set. In this case the internal counter clears as soon as the match signal occurs. The channel FLAG is set at the same time the match occurs. Note that by having the internal counter cleared as soon as the match occurs and incremented at the next input event a shorter zero count is generated. See [Figure 36-164](#) and [Figure 36-165](#).
- Internal clock source is selected if MODE[0] is cleared. In this case the counter clears as soon as the match signal occurs. The channel FLAG is set at the same time the match occurs. At the next prescaler tick after the match the internal counter remains at zero and only resumes counting on the following tick. See [Figure 36-164](#) and [Figure 36-166](#).
- Internal counter clearing on match end (MODE[0:6]=001001b)
 - External clock is selected if MODE[0] is set. In this case the internal counter clears when the match signal is asserted and the input event occurs. The channel FLAG is set at the same time the counter is cleared. See [Figure 36-164](#) and [Figure 36-167](#).
 - Internal clock source is selected if MODE[0] is cleared. In this case the internal counter clears when the match signal is asserted and the prescaler tick occurs. The channel FLAG is set at the same time the counter is cleared. See [Figure 36-164](#) and [Figure 36-167](#).

Note

If internal clock source is selected and the prescaler of the internal counter is set to 1 the MC mode behaves the same way even in Clear on Match Start or Clear on Match End sub-modes.

When in up/down count mode (MODE[0:6]=00101bb), a match between the internal counter and register A1 sets the FLAG and changes the counter direction from increment to decrement. A match between register B1 and the internal counter changes the counter direction from decrement to increment and sets the FLAG only if MODE[5] bit is set.

Only values different than 0x0 must be written at A register. Loading 0x0 leads to unpredictable results.

Updates on register A or counter in MC mode may cause loss of match in the current cycle if the transfer occurs near the match. In this case, the counter may rollover and resume operation in the next cycle.

Register B2 has no effect in MC mode. Nevertheless, register B2 can be accessed for reads and writes by addressing EMIOB.

The following two figures show how the Unified Channel can be used as modulus counter in up mode and up/down mode, respectively.

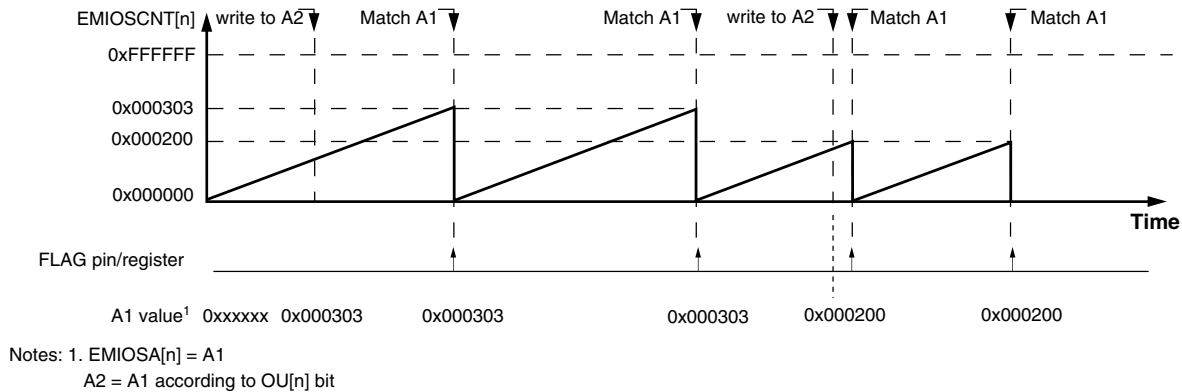


Figure 36-135. Modulus Counter up mode example (MODE[4] bit is 0)

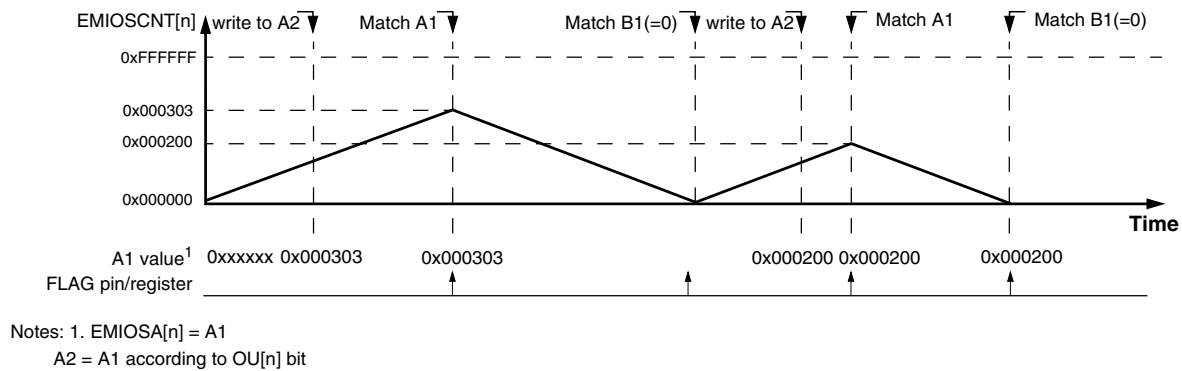


Figure 36-136. Modulus Counter up/down mode example (MODE[4] bit is 1)

36.6.1.1.12 Modulus Counter Buffered (MCB) Mode

The MCB mode provides a time base which can be shared with other channels through the internal counter buses. Register A1 is double buffered thus allowing smooth transitions between cycles when changing A2 register value on the fly. Register A1 is updated at the cycle boundary, which is defined as when the internal counter transitions to 0x1.

The internal counter values operates within a range from 0x1 up to register A1 value. If when entering MCB mode coming out from GPIO mode the internal counter value is not within that range then the A match will not occur causing the channel internal counter to wrap at the maximum counter value which is 0xff_ff for a 24-bit counter. After the counter wrap occurs it returns to 0x1 and resume normal MCB mode operation. Thus in order to avoid the counter wrap condition make sure its value is within the 0x1 to register A1 value range when the MCB mode is entered.

MODE[6] bit selects internal clock source if cleared or external if set. When external clock is selected the input channel pin is used as the channel clock source. The active edge of this clock is defined by EDPOL and EDSEL bits in the EMIOSC[n] channel register.

When entering in MCB mode, if up counter is selected by MODE[4]=0 (MODE[0:6]=101000b), the internal counter starts counting from its current value to up direction until A1 match occurs. The internal counter is set to 0x1 when its value matches A1 value and a clock tick occurs (either prescaled clock or input pin event).

If up/down counter is selected by setting MODE[4]=1, the counter changes direction at A1 match and counts down until it reaches the value 0x1. After it has reached 0x1 it is set to count in up direction again. Register B1 is used to generate a match in order to set the internal counter in up-count direction if up/down mode is selected. Register B1 cannot be changed while this mode is selected.

Note that differently from the MC mode, the MCB mode counts between 0x1 and A1 register value. Only values greater than 0x1 must be written at register A1. Loading values other than those leads to unpredictable results. The counter cycle period is equal to A1 value in up counter mode. If in up/down counter mode the period is defined by the expression: $(2 \cdot A1) - 2$.

Figure 36-137 describes the counter cycle for several A1 values. Register A1 is loaded with A2 register value at the cycle boundary. Thus any value written to A2 register within cycle **n** will be updated to A1 at the next cycle boundary and therefore will be used on cycle **n+1**. The cycle boundary between cycle **n** and cycle **n+1** is defined as when the internal counter transitions from A1 value in cycle **n** to 0x1 in cycle **n+1**. Note that the FLAG is generated at the cycle boundary and has a synchronous operation, meaning that it is asserted one system clock cycle after the FLAG set event.

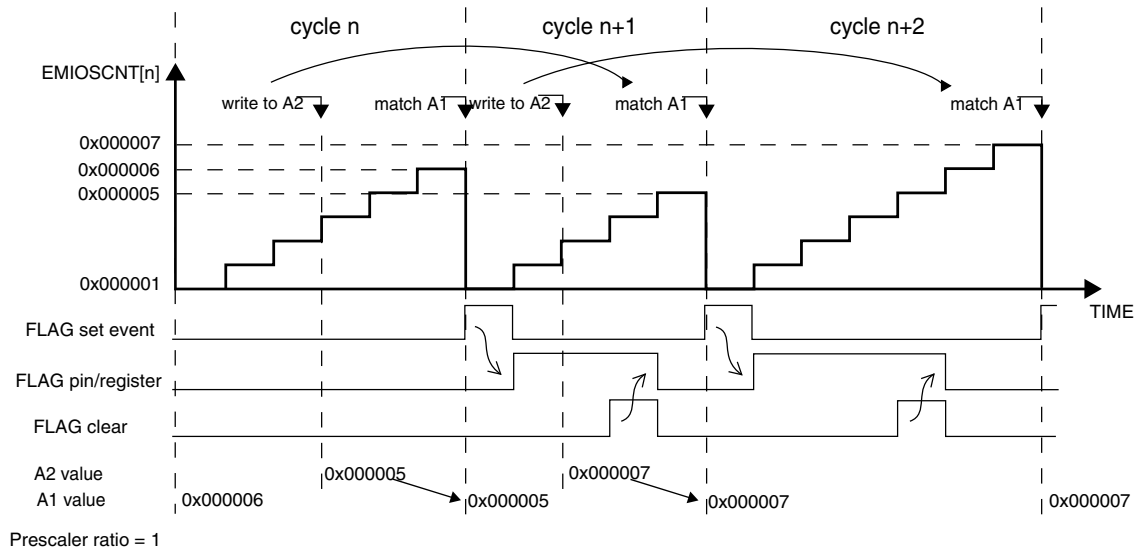


Figure 36-137. Modulus Counter Buffered (MCB) Up Count mode

Figure 36-138 describes the MCB in up/down counter mode (MODE[0:6]=10101bb). Register A1 is updated at the cycle boundary. If A2 is written in cycle *n*, this new value will be used in cycle *n+1* for A1 match. Flags are generated only at A1 match start if MODE[5] is 0. If MODE[5] is set to 1 flags are also generated at the cycle boundary.

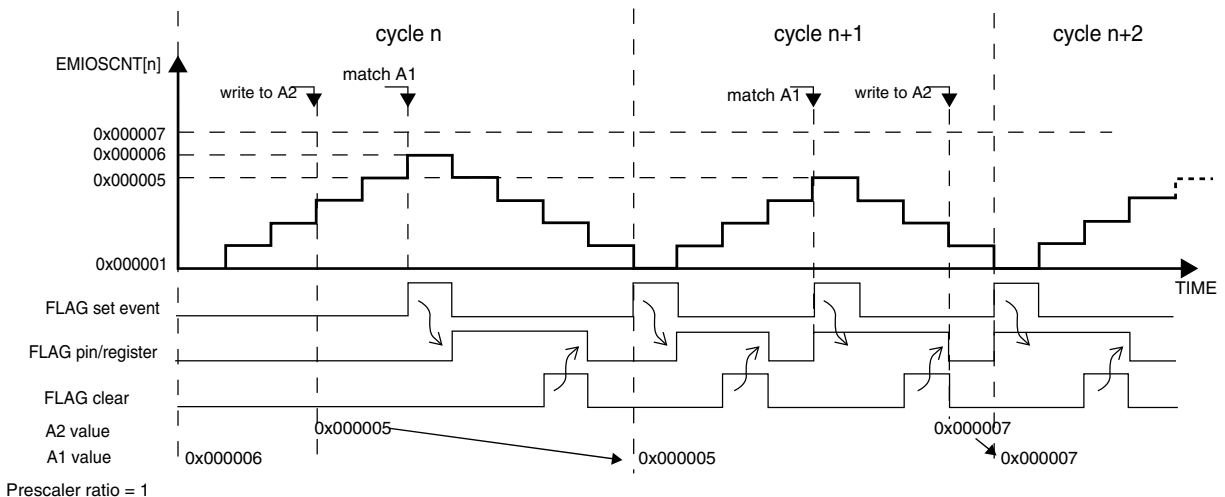


Figure 36-138. Modulus Counter Buffered (MCB) Up/Down Mode

The figure below describes in more detail, the register A1 update process in up counter mode. The A1 load signal is generated at the last system clock period of a counter cycle. Thus, A1 is updated with A2 value at the same time that the counter (EMIOSCNT[n]) is loaded with 0x1. The load signal pulse has the duration of one system clock period. If A2 is written within cycle *n* its value is available at A1 at the first clock of cycle *n+1* and the new value is used for match at cycle *n+1*. The update disable bits OU[n] of EMIOSOUDIS register can be used to control the update of this register, thus allowing to delay the A1 register update for synchronization purposes.

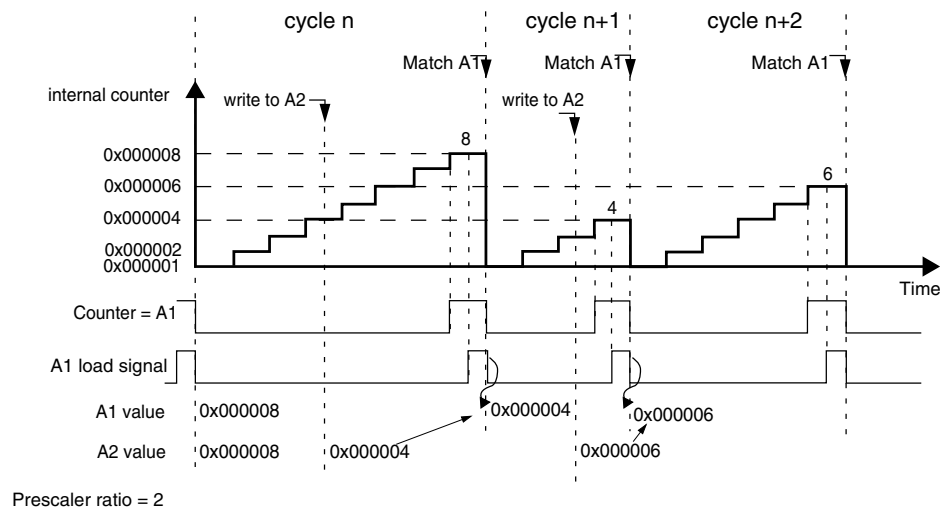


Figure 36-139. MCB Mode Register A1 Update in Up Counter Mode

The figure below describes the register A1 update in up/down counter mode. Note that A2 can be written at any time within cycle **n** in order to be used in cycle **n+1**. Thus A1 receives this new value at the next cycle boundary. Note that the update disable bits **OU[n]** of **EMIOSOUDIS** register can be used to disable the update of register A1.

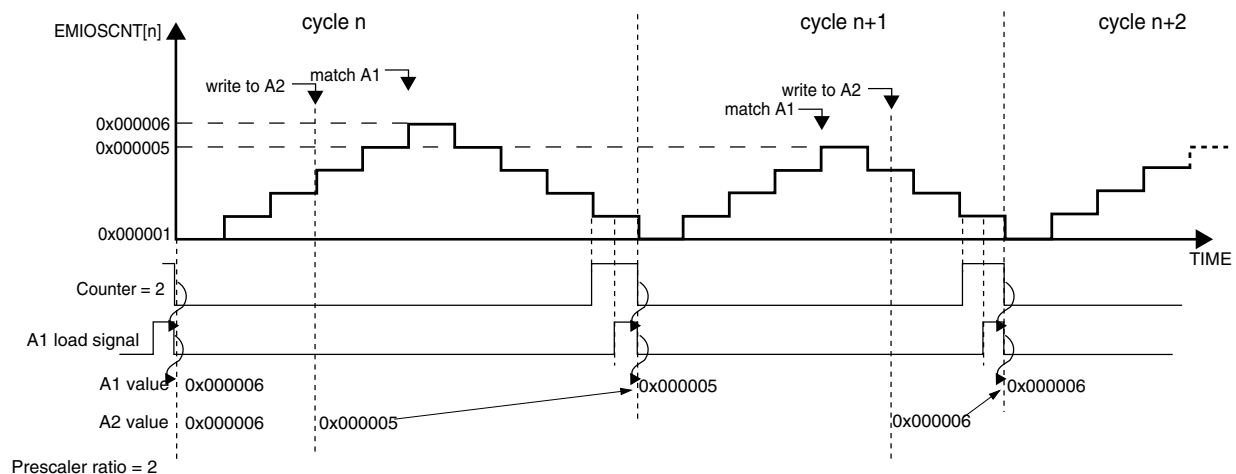


Figure 36-140. MCB Mode Register A1 Update in Up/Down Counter Mode

36.6.1.1.13 Output Pulse Width and Frequency Modulation (OPWFM) Mode

In this mode, duty cycle of output signal is the value defined in register A1 plus one and the period is the value defined in register B1 plus one. **MODE[6]** bit controls the transfer from register B2 to B1, which can be done either immediately (**MODE[6]** cleared, **MODE[0:6]=00110b0**), providing the fastest change in the duty cycle, or at every match of register A1 (**MODE[6]** set, **MODE[0:6]=00110b1**).

When OPWFM mode is entered, coming out from GPIO mode, the output flip-flop is set to the complement of the **EDPOL** bit in the **EMIOSC[n]** register.

The internal counter is automatically selected as a time base, therefore the BSL[0:1] bits in register EMIOSC[n] has no meaning. When a match on comparator A occurs, the output flip-flop is set to the value of the EDPOL bit. When a match occurs on comparator B, the output flip-flop is set to the complement of the EDPOL bit and the internal counter is cleared.

A FLAG can be generated at match B, when MODE[5] is cleared, or in both matches, when MODE[5] is set.

At any time, the FORCMA and FORCMB bits allow the software to force the output flip-flop to the level corresponding to a match on A or B respectively. Also, FORCMB clears the internal counter. Note that the FLAG bit is not set by the FORCMA or FORCMB operations.

If subsequent comparisons occur on comparators A and B, the PWFM pulses continue to be output, regardless of the state of the FLAG bit.

In order to achieve 100% duty cycle, both registers A1 and B1 must be set to the same value. When a simultaneous match occurs on comparators A and B, the output flip-flop is set to the value of EDPOL bit. 0% duty cycle is possible by writing 0x0 to register A. When a match occurs, the output flip-flop is set at every period to the complement of EDPOL bit. The transfer from register B2 to B1 is still controlled by MODE[6] bit.

Note

Writing 0x0 to A1 and B1 produces a duty cycle of 0%.

Figure 36-141 shows the Unified Channel running in OPWFM mode with immediate register update and Figure 36-142 shows the Unified Channel running in OPWFM mode with next period update.

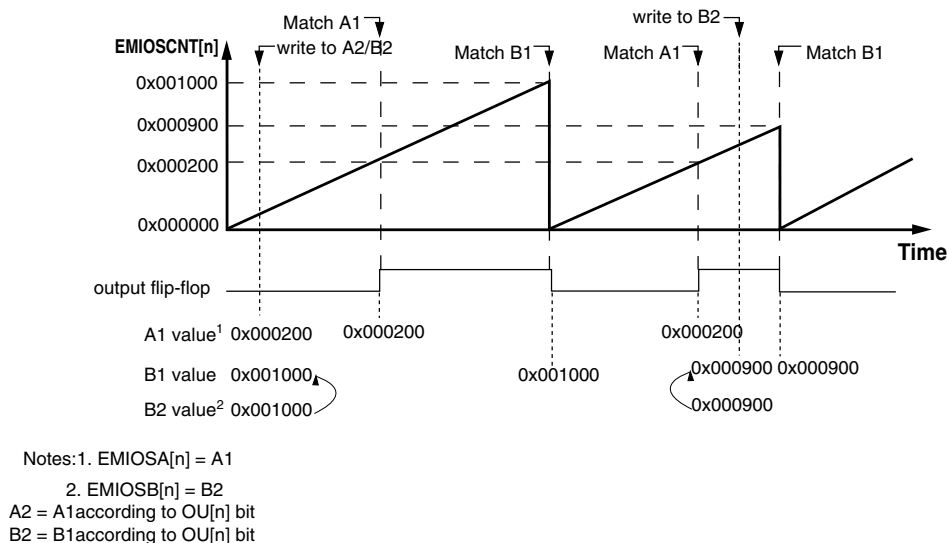


Figure 36-141. OPWFM with immediate update (MODE[6] bit is 0b)

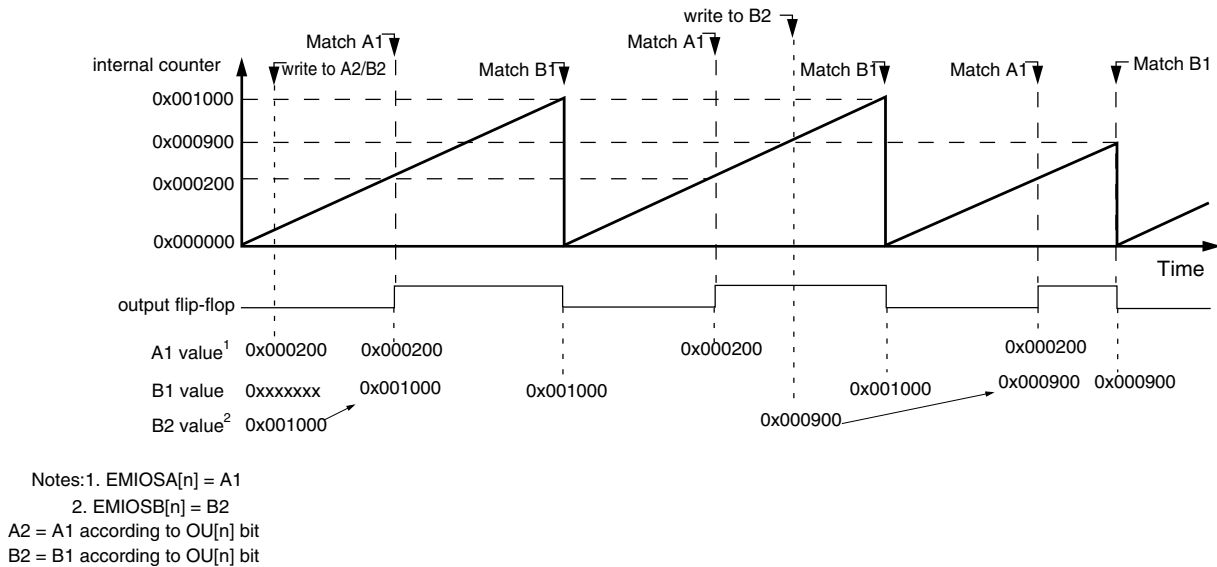


Figure 36-142. OPWFM with next period update (MODE[6] bit is 1b)

36.6.1.1.14 Output Pulse Width and Frequency Modulation Buffered (OPWFMB) Mode

This mode (MODE[0:6]=10110b0) provides waveforms with variable duty cycle and frequency. The internal channel counter is automatically selected as the time base when this mode is selected. Register A1 indicates the duty cycle and B1 register the frequency. Both registers A1 and B1 are double buffered to allow smooth signal generation when changing the registers values.. 0% and 100% duty cycles are supported.

At OPWFMB mode entry, the output flip-flop is set to the value of the EDPOL bit in the EMIOSC[n] register.

In order to provide smooth and consistent channel operation this mode differs substantially from the OPWFM mode. The main differences reside in the A1 and B1 registers update, on the delay from the A1 match to the output pin transition and on the range of the internal counter values which starts from 0x1 up to register B1 value.

When entering OPWFMB mode coming out of GPIO mode, the internal counter value is not within that range then the B match will not occur causing the channel internal counter to wrap at the maximum counter value which is 0xff_fff for a 24-bit counter. After the counter wrap occurs it returns to 0x1 and resume normal OPWFMB mode operation. Thus in order to avoid the counter wrap condition make sure its value is within the 0x1 to B1 register value range when the OPWFMB mode is entered.

When a match on comparator A occurs the output register is set to the value of EDPOL. When a match on comparator B occurs the output register is set to the complement of EDPOL. B1 match also causes the internal counter to transition to 0x1, thus restarting the counter cycle.

Only values greater than 0x1 are allowed to be written to register B1. Loading values other than those leads to unpredictable results.

Figure 36-143 describes the operation of the OPWFMB mode regarding output pin transitions and A1/B1 registers match events. Note that the output pin transition occurs when the A1 or B1 match signal is deasserted which is indicated by the A1 match negedge detection signal. If register A1 is set to 0x4 the output pin transitions 4 counter periods after the cycle had started, plus one system clock cycle. Note that in the example shown in Figure 36-143 the internal counter prescaler has a ratio of two.

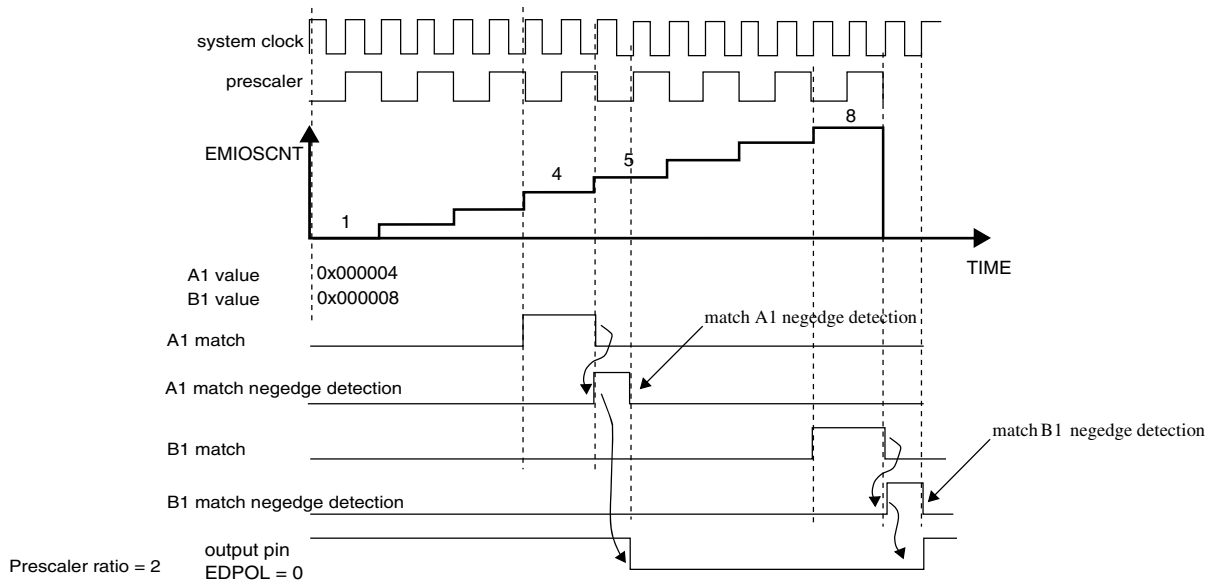


Figure 36-143. OPWFMB A1 and B1 match to Output Register Delay

Figure 36-144 describes the generated output signal if A1 is set to 0x0. Since the counter does not reach zero in this mode, the channel internal logic infers a match as if A1=0x1 with the difference that in this case, the posedge of the match signal is used to trigger the output pin transition instead of the negedge used when A1=0x1. Note that A1 posedge match signal from cycle $n+1$ occurs at the same time as B1 negedge match signal from cycle n . This allows to use the A1 posedge match to mask the B1 negedge match when they occur at the same time. The result is that no transition occurs on the output flip-flop and a 0% duty cycle is generated.

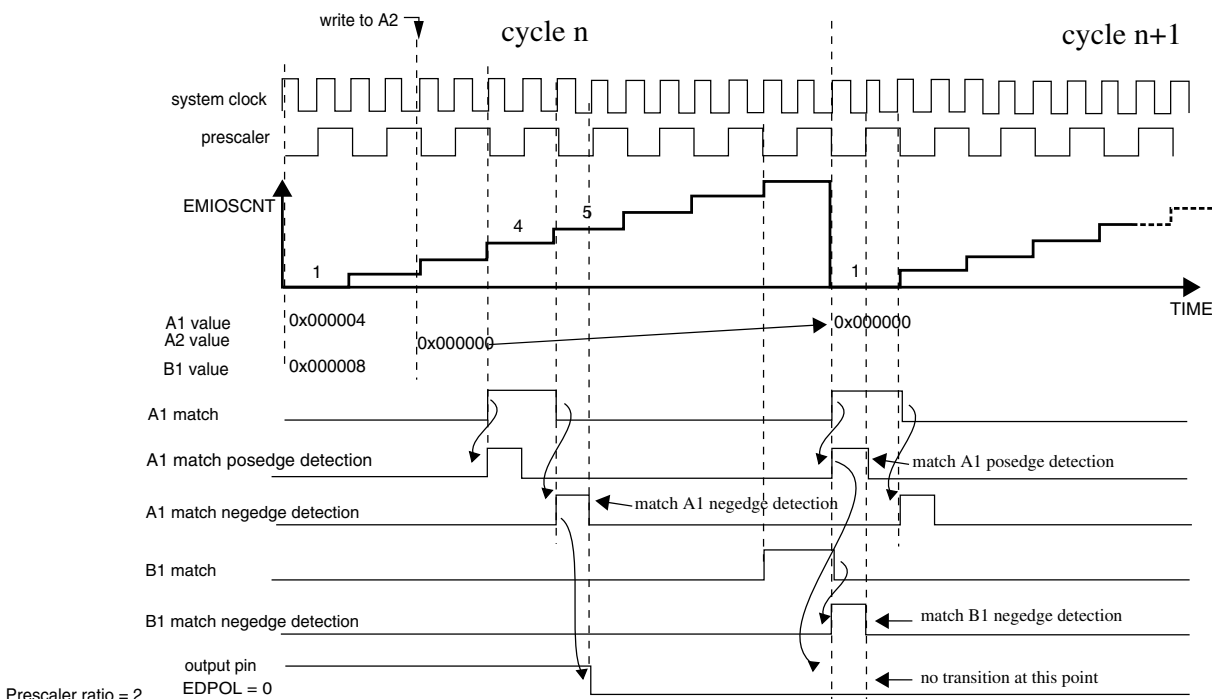


Figure 36-144. OPWFMB Mode with A1 = 0 (0% duty cycle)

Figure 36-145 describes the timing for the A1 and B1 registers load. The A1 and B1 load use the same signal which is generated at the last system clock period of a counter cycle. Thus, A1 and B1 are updated respectively with A2 and B2 values at the same time that the counter (EMIOSCNT[n]) is loaded with 0x1. This event is defined as the cycle boundary. The load signal pulse has the duration of one system clock period. If A2 and B2 are written within cycle **n** their values are available at A1 and B1, respectively, at the first clock of cycle **n+1** and the new values are used for matches at cycle **n+1**. The update disable bits OU[n] of EMIOSOUDIS register can be used to control the update of these registers, thus allowing to delay the A1 and B1 registers update for synchronization purposes.

In Figure 36-145, it is assumed that both the channel and global prescalers are set to 0x1 (each divide ratio is two), meaning that the channel internal counter transitions at every four system clock cycles. FLAGS can be generated only on B1 matches when MODE[5] is cleared, or on both A1 and B1 matches when MODE[5] is set. Since B1 flag occurs at the cycle boundary, this flag can be used to indicate that A2 or B2 data written on cycle **n** were loaded to A1 or B1, respectively, thus generating matches in cycle **n+1**. Note that the FLAG has a synchronous operation, meaning that it is asserted one system clock cycle after the FLAG set event.

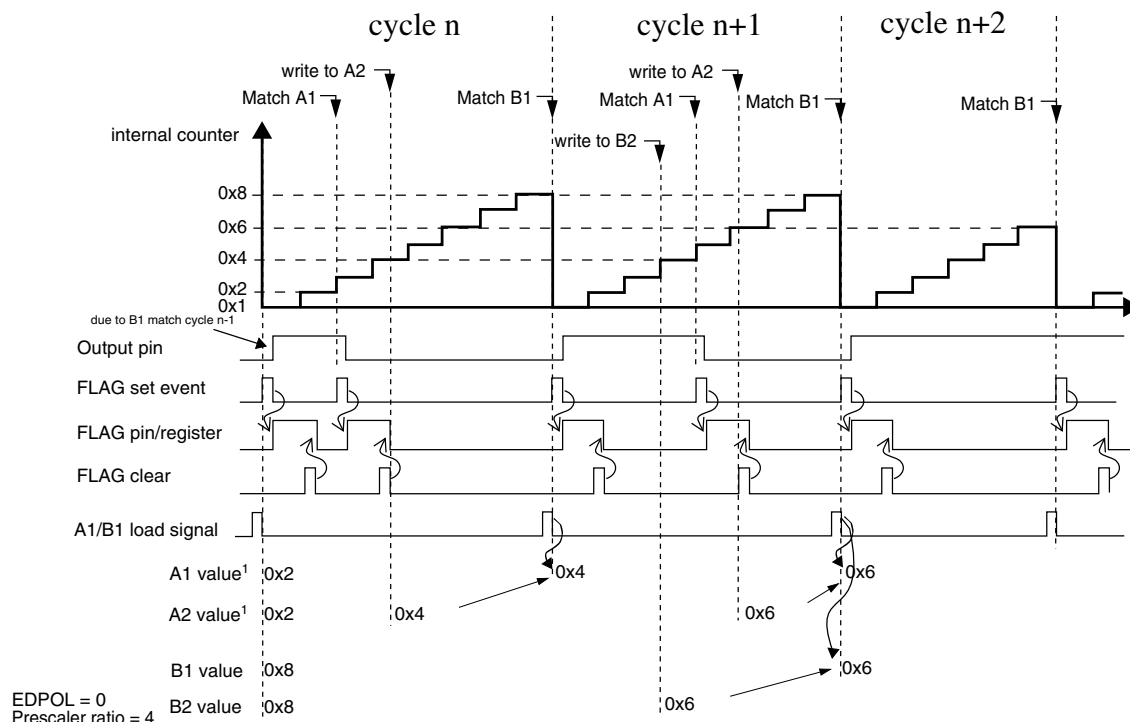


Figure 36-145. OPWFMB A1 and B1 Registers Update and Flags (MODE6] bit is 0b)

Figure 36-146 describes the operation of the Output Disable feature in OPWFMB mode. Differently from the OPWFM mode, the output disable forces the channel output flip-flop to EDPOL bit value. This functionality targets applications that use active high signals and a high to low transition at A1 match. In this case EDPOL should be set to 0. Note that both the channel and global prescalers are set to 0x0 (each divide ratio is one), meaning that the channel internal counter transitions at every system clock cycle.

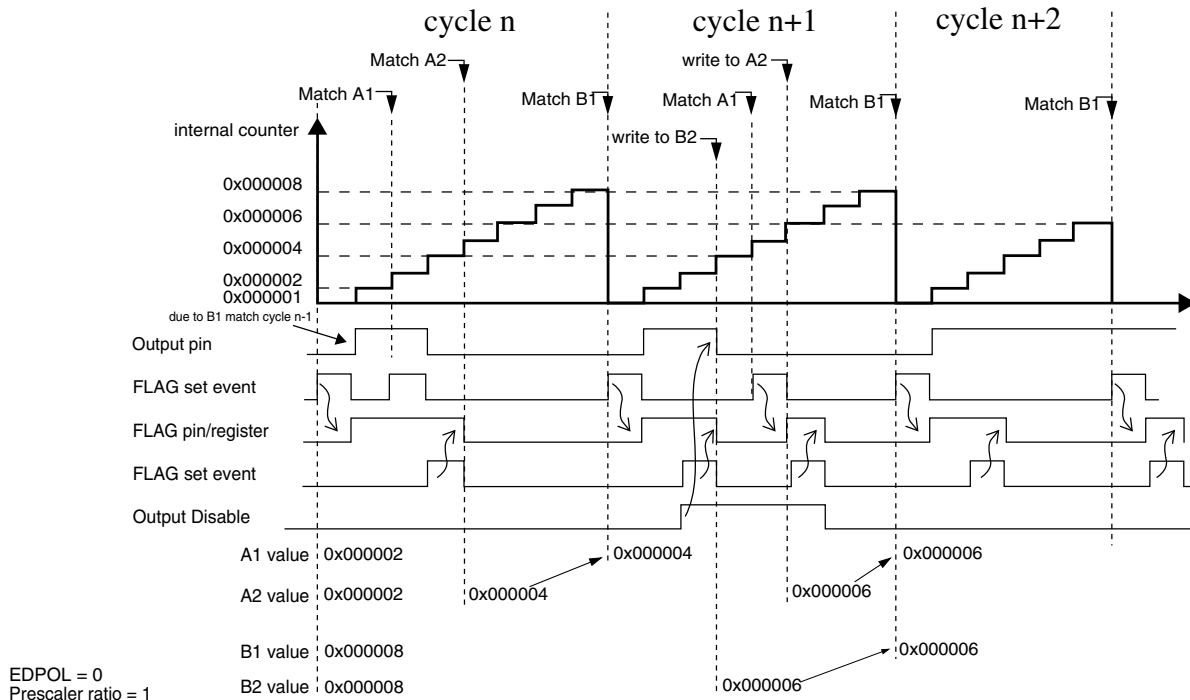


Figure 36-146. OPWFMB Mode with Active Output Disable (MODE[6] bit is 1b)

Note that the output disable has a synchronous operation, meaning that the assertion of the Output Disable input pin causes the channel output flip-flop to transition to EDPOL at the next system clock cycle. If the Output Disable input is deasserted the output pin transition at the following A1 or B1 match.

In [Figure 36-146](#), it is assumed that the Output Disable input is enabled and selected for the Channel. Please, refer to UC Control Register (EMIOSCn) for a detailed description about the ODIS and ODISSL bits, respectively enable and selection of the Output Disable inputs.

The FORCMA and FORCMB bits allow the software to force the output flip-flop to the level corresponding to a match on comparators A or B respectively. Similarly to a B1 match FORCMB sets the internal counter to 0x1. The FLAG bit is not set by the FORCMA or FORCMB bits being asserted.

[Figure 36-147](#) describes the generation of 100% and 0% duty cycle signals. It is assumed $EDPOL = 0$ and the resultant prescaler value is 1. Initially, $A1 = 0x8$ and $B1 = 0x8$. In this case, B1 match has precedence over A1 match, thus the output flip-flop is set to the complement of EDPOL bit. This cycle corresponds to a 100% duty cycle signal. The same output signal can be generated for any A1 value greater or equal to B1.

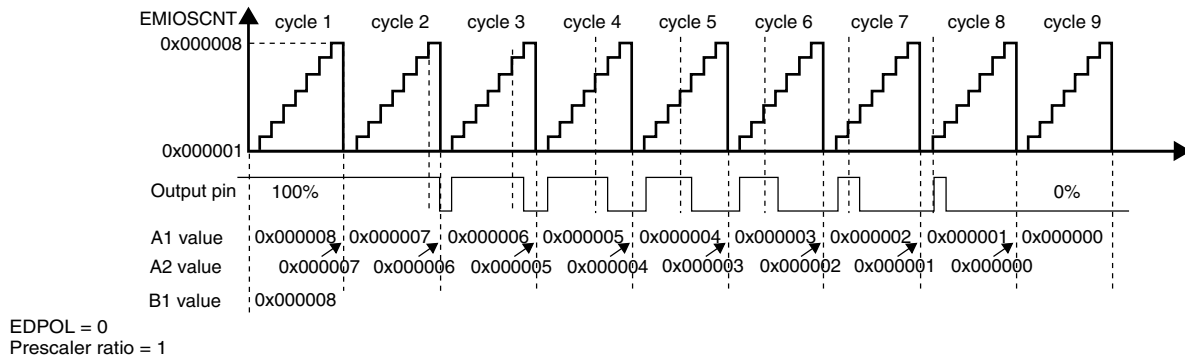


Figure 36-147. OPWFMB Mode from 100% to 0% Duty Cycle

A 0% duty cycle signal is generated if A1=0x0 as shown in [Figure 36-147](#) cycle 9. In this case B1=0x8 match from cycle 8 occurs at the same time as the A1=0x0 match from cycle 9. Please, refer to [Figure 36-144](#) for a description of the A1 and B1 match generation. In this case A1 match has precedence over B1 match and the output signal transitions to EDPOL.

36.6.1.1.15 Center Aligned Output Pulse Width Modulation with Dead Time Insertion (OPWMC) Mode

This operation mode generates a center aligned PWM with dead time insertion in the leading (MODE[0:6]=00111b1) or trailing edge (MODE[0:6]=00111b0).

The selected counter bus must be running an up/down time base, as shown in the 'Modulus Counter up/down mode example' figure. BSL[0:1] bits select the time base. Register A1 contains the ideal duty cycle for the PWM signal and is compared with the selected time base. Register B1 contains the dead time value and is compared with the internal counter. For a leading edge dead time insertion, the output PWM duty cycle is equal to the difference between register A1 and register B1, and for a trailing edge dead time insertion, the output PWM duty cycle is equal to the sum of register A1 and register B1. Mode[6] bit selects between trailing and leading dead time insertion, respectively.

Note

The internal counter may be running in the internal prescaler ratio, while the selected time base may be running in a different prescaler ratio. The output signal may produce an unexpected output if the dead time interval is greater than the duty cycle of the PWM signal.

When OPWMC mode is entered, coming out from GPIO mode, the output flip-flop is set to the complement of the EDPOL bit in the EMIOSC[n] register.

When operating with leading edge dead time insertion, the first match between A1 and the selected time base clears the internal counter and switches the selected time base to the internal counter. When a match occurs between register B1 and the selected time base, the output flip-flop is set to the value of the EDPOL bit and the time base is switched to the selected counter bus. In the next match between register A1 and the selected time base, the output flip-flop is set to the complement of the EDPOL bit. This sequence repeats continuously.

When operating with trailing edge dead time insertion, the first match between A1 and the selected time base sets the output flip-flop to the value of the EDPOL bit. In the next match between register A1 and the selected time base, the internal counter is cleared and the selected time base is switched to the internal counter. When a match occurs between register B1 and the selected time base, the output flip-flop is set to the complement of the EDPOL bit and the time base is switched to the selected counter bus. This sequence repeats continuously.

FLAG can be generated in the trailing edge of the output PWM signal when MODE[5] is cleared, or in both edges, when MODE[5] is set.

At any time, the FORCMA or FORCMB bits are equivalent to a successful comparison on comparator A or B with the exception that the FLAG bit is not set.

Note

When in freeze state, the FORCMA or FORCMB bits only allow the software to force the output flip-flop to the level corresponding of a match on A or B respectively.

If subsequent matches occur on comparators A and B, the PWM pulses continue to be generated, regardless of the state of the FLAG bit.

In order to achieve a duty cycle of 100%, both registers A1 and B1 must be set to the same value. When a simultaneous match occurs between the selected time base and registers A1 and B1, the output flip-flop is set to the value of EDPOL bit and the selected time base switches to the selected counter bus, allowing a new cycle to begin at any time, as previously described. 0% duty cycle is possible by writing 0x0 to register A. When a match occurs, the output flip-flop is set to the complement of EDPOL bit and the selected time base switches to the selected counter bus, allowing a new cycle to begin at any time, as previously described. In both cases, FLAG is generated regardless of MODE[6] bit.

Note

If A1 and B1 are set to the 0x0, a 0% duty cycle waveform is produced.

[Figure 36-148](#) and [Figure 36-149](#) show the Unified Channel running in OPWMC with leading and trailing dead time, respectively.

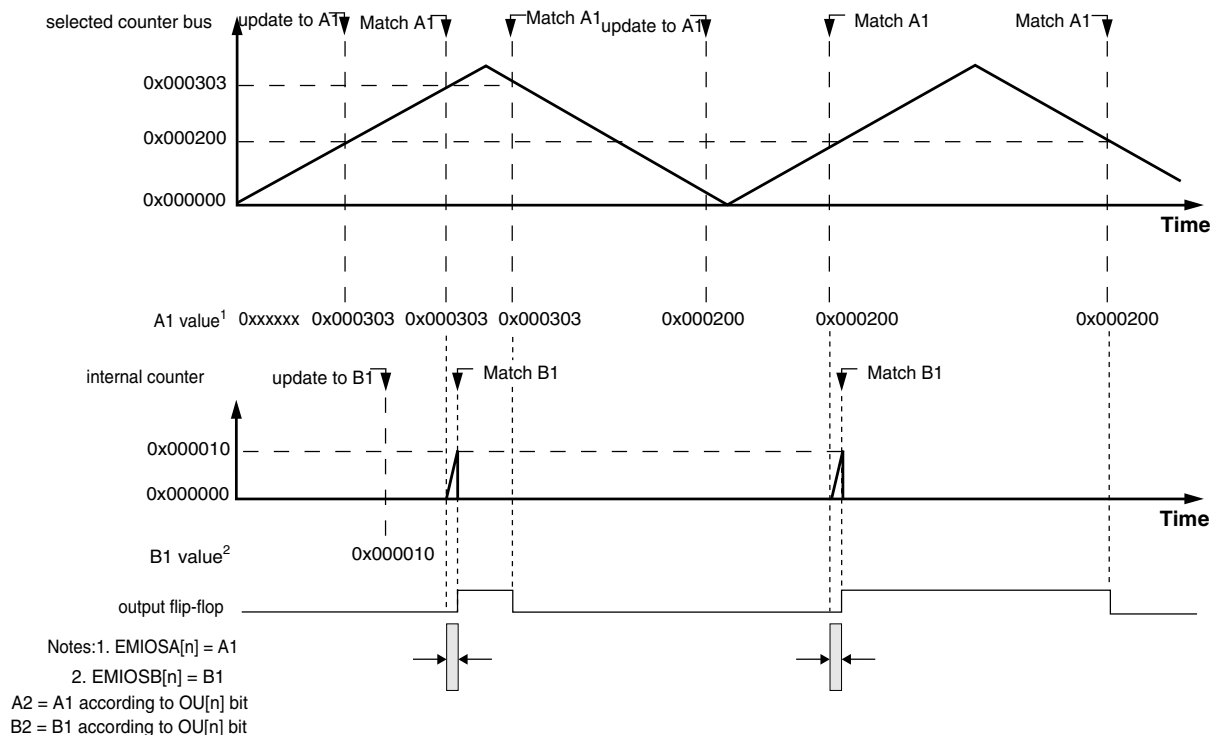


Figure 36-148. OPWMC with leading dead time insertion (MODE [6] bit is 1b)

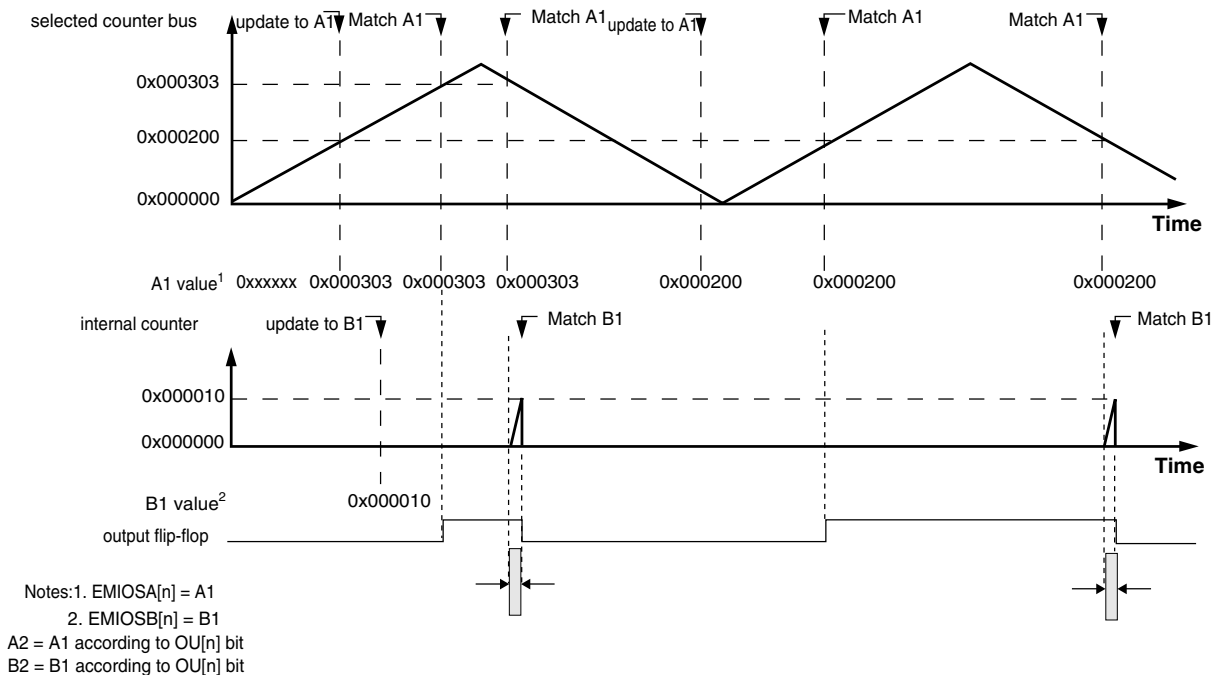


Figure 36-149. OPWMC with trailing dead time insertion (MODE [6] bit is 0b)

36.6.1.1.16 Center Aligned Output Pulse Width Modulation with Dead Time Insertion Buffered (OPWMCB) Mode

This operation mode generates a center aligned PWM with dead time insertion to the leading (MODE[0:6]=10111b1) or trailing edge (MODE[0:6]=10111b0). A1 and B1 registers are double buffered to allow smooth output signal generation when changing A2 or B2 registers values.

BSL[0:1] bits select the time base. The time base selected for a channel configured to OPWMCB mode should be a channel configured to MCB Up/Down mode, as shown in [Figure 36-138](#). It is recommended to start the MCB channel time base after the OPWMCB mode is entered in order to avoid missing A matches at the very first duty cycle.

Register A1 contains the ideal duty cycle for the PWM signal and is compared with the selected time base.

Register B1 contains the dead time value and is compared against the internal counter. For a leading edge dead time insertion, the output PWM duty cycle is equal to the difference between register A1 and register B1, and for a trailing edge dead time insertion, the output PWM duty cycle is equal to the sum of register A1 and register B1. Mode[6] bit selects between trailing and leading dead time insertion, respectively.

Note

The internal counter runs in the internal prescaler ratio, while the selected time base may be running in a different prescaler ratio.

When OPWMCB mode is entered, coming out from GPIO mode, the output flip-flop is set to the complement of the EDPOL bit in the EMIOSC[n] register.

The following basic steps summarize proper OPWMCB startup, assuming the channels are initially in GPIO mode:

1. *[global]* Disable Global Prescaler.
2. *[MCB channel]* Disable Channel Prescaler.
3. *[MCB channel]* Write 0x1 at internal counter.
4. *[MCB channel]* Set A register.
5. *[MCB channel]* Set channel to MCB Up mode.
6. *[MCB channel]* Set prescaler ratio.
7. *[MCB channel]* Enable Channel Prescaler.

8. [OPWMCB channel] Disable Channel Prescaler.
9. [OPWMCB channel] Set A register.
10. [OPWMCB channel] Set B register.
11. [OPWMCB channel] Select time base input through BSL[1:0] bits.
12. [OPWMCB channel] Enter OPWMCB mode.
13. [OPWMCB channel] Set prescaler ratio.
14. [OPWMCB channel] Enable Channel Prescaler.
15. [global] Enable Global Prescaler.

The following figure describes the load of A1 and B1 registers which occurs when the selected counter bus transitions from 0x2 to 0x1. This event defines the cycle boundary. Note that values written to A2 or B2 within cycle **n** are loaded into A1 or B1 registers, respectively, and used to generate matches in cycle **n+1**.

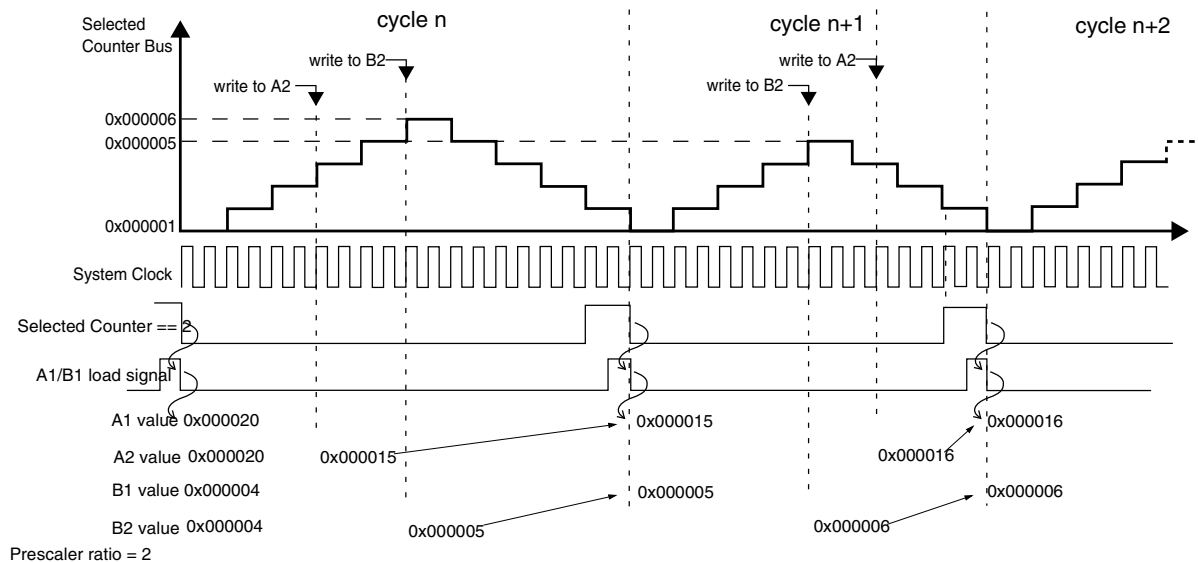


Figure 36-150. OPWMCB A1 and B1 registers load

The OU[n] bit of EMIOSOUDIS register can be used to disable the A1 and B1 updates, thus allowing to synchronize the load on these registers with the load of A1 or B1 registers in others channels. Note that using the update disable bit A1 and B1 registers can be updated at the same counter cycle thus allowing to change both registers at the same time.

In this mode, A1 matches always sets the internal counter to 0x1. When operating with leading edge dead time insertion the first A1 match sets the internal counter to 0x1. When a match occurs between register B1 and the internal time base, the output flip-flop is set

to the value of the EDPOL bit. In the following match between register A1 and the selected time base, the output flip-flop is set to the complement of the EDPOL bit. This sequence repeats continuously. The internal counter should not reach 0x0 as consequence of a rollover. In order to avoid it, the user should not write to EMIOSB register a value greater than twice the difference between external count up limit and EMIOSA value.

The following figure shows two cycles of a Center Aligned PWM signal. Note that both A1 and B1 register values are changing within the same cycle which allows to vary at the same time the duty cycle and dead time values.

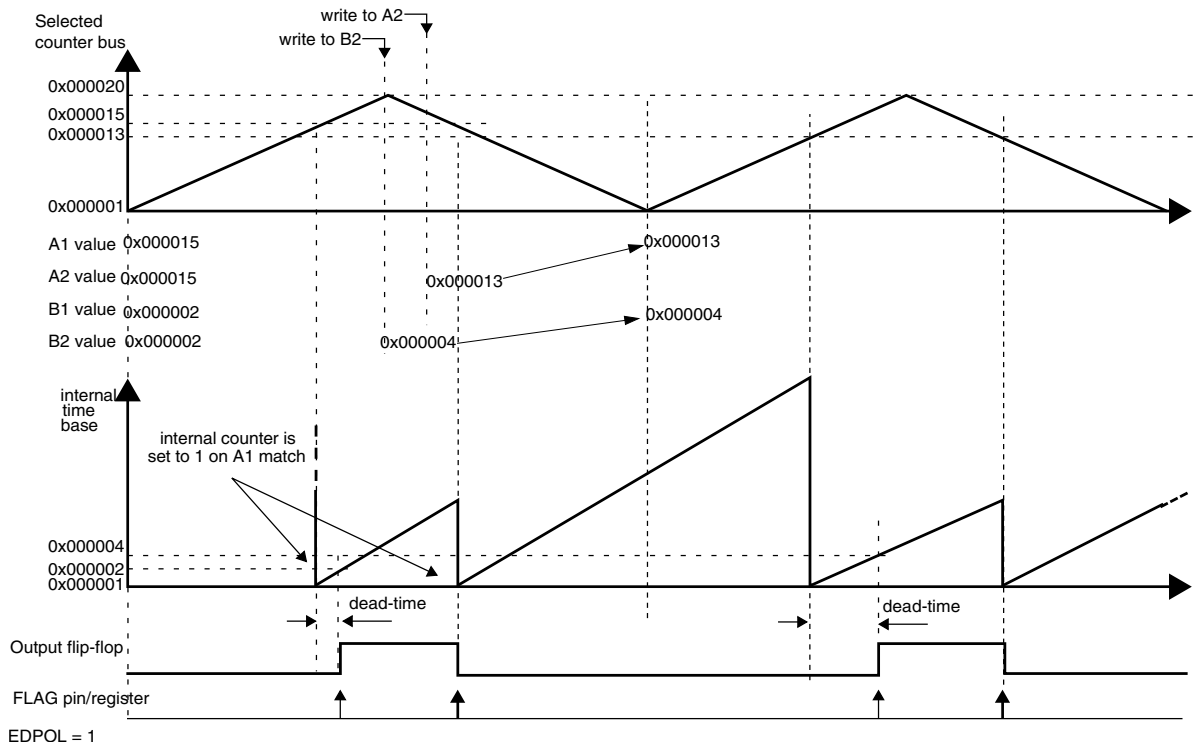


Figure 36-151. OPWMCB with Lead Dead Time Insertion

When operating with trailing edge dead time insertion, the first match between A1 and the selected time base sets the output flip-flop to the value of the EDPOL bit and sets the internal counter to 0x1. In the second match between register A1 and the selected time base, the internal counter is set to 0x1 and B1 matches are enabled. When the match between register B1 and the selected time base occurs the output flip-flop is set to the complement of the EDPOL bit. This sequence repeats continuously.

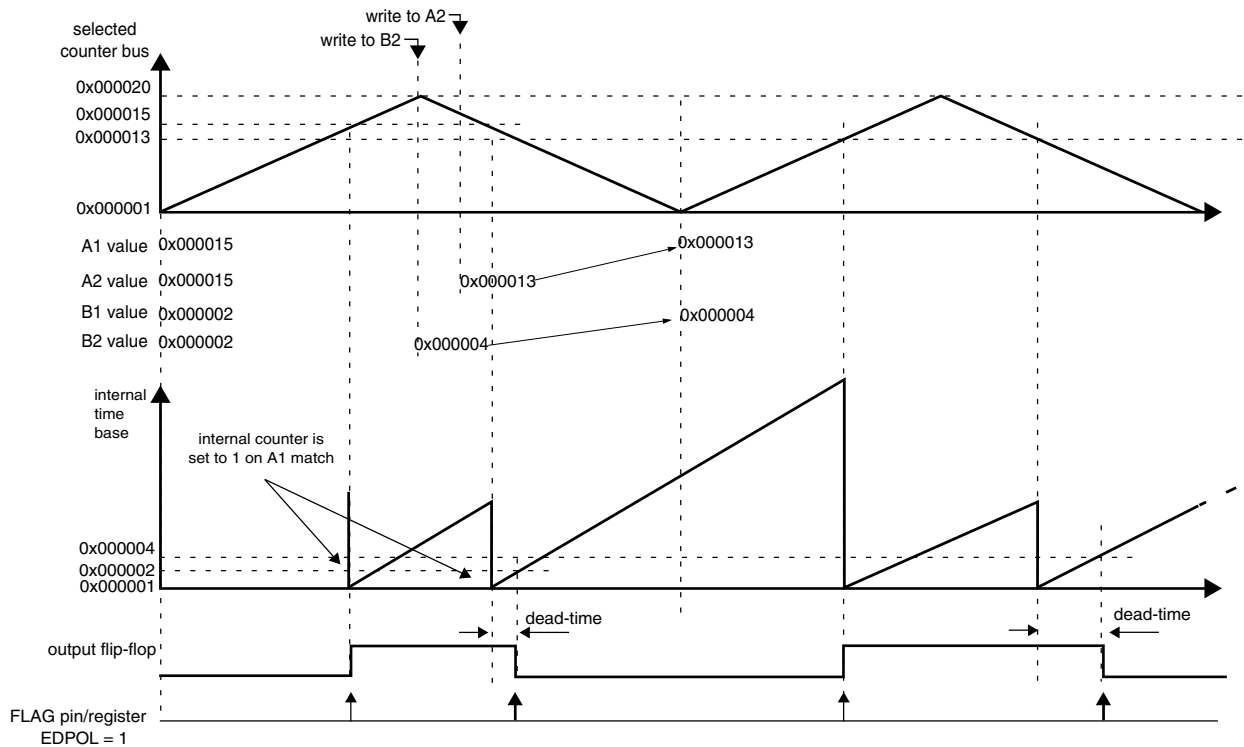


Figure 36-152. OPWMCB with Trail Dead Time Insertion

A FLAG can be generated in the trailing edge of the output PWM signal when MODE[5] is cleared, or in both edges, when MODE[5] is set. If subsequent matches occur on comparators A and B, the PWM pulses continue to be generated, regardless of the state of the FLAG bit.

Note

In OPWMCB mode FORCMA and FORCMB do not have the same behavior as a regular match. Instead they force the output flip-flop to constant value which depends upon the selected dead time insertion mode, lead or trail and the value of the EDPOL bit.

FORCMA has different behaviors depending upon the selected dead time insertion mode, lead or trail. In lead dead time insertion FORCMA force a transition in the output flip-flop to the opposite of EDPOL. In trail dead time insertion the output flip-flop is forced to the value of EDPOL bit.

If FORCMB bit is set, the output flip-flop value depends upon the selected dead time insertion mode. In lead dead time insertion FORCMB forces the output flip-flop to transition to EDPOL bit value. In trail dead time insertion the output flip-flop is forced to the opposite of EDPOL bit value.

Note

FORCMA bit set does not set the internal time-base to 0x1 as a regular A1 match.

The FLAG bit is not set either in case of a FORCMA or FORCMB or even if both forces are issued at the same time.

Note

FORCMA and FORCMB have the same behavior even in Freeze or normal mode regarding the output pin transition.

When *FORCMA* is issued along with *FORCMB* the output flip-flop is set to the opposite of *EDPOL* bit value. This is equivalent of saying that *FORCMA* has precedence over *FORCMB* when lead dead time insertion is selected and *FORCMB* has precedence over *FORCMA* when trail dead time insertion is selected.

A duty cycle from 0% to 100% can be generated by setting appropriate values to A1 and B1 registers relatively to the period of the external time base. Setting A1=0x1 generates a 100% duty cycle waveform. Assuming *EDPOL* is set to one and *OPWMCB* mode with trail dead time insertion, 100% duty cycle signals can be generated if B1 occurs at or after the cycle boundary (external counter = 1). If A1 is greater than the maximum value of the selected counter bus period, then a 0% duty cycle is produced, only if the pin starts the current cycle in the opposite of *EDPOL* value. In case of 100% duty cycle, the transition from *EDPOL* to the opposite of *EDPOL* may be obtained by forcing pin, using *FORCMA* and/or *FORCMB*.

Note

If A1 is set to 0x1 at *OPWMCB* entry the 100% duty cycle may not be obtained in the very first PWM cycle due to the pin condition at mode entry.

Only values different than 0x0 are allowed to be written to A1 register. If 0x0 is loaded to A1, the results are unpredictable.

Note

A special case occurs when A1 is set to (external counter bus period)/2, which is the maximum value of the external counter. In this case the output flip-flop is constantly set to the *EDPOL* bit value.

The internal channel logic prevents matches from one cycle to propagate to the next cycle. In trail dead time insertion B1 match from cycle **n** could eventually cross the cycle boundary and occur in cycle **n+1**. In this case B1 match is masked out and does not cause the output flip-flop to transition. Therefore matches in cycle **n+1** are not affected by the late B1 matches from cycle **n**.

The following figure shows a 100% duty cycle output signal generated by setting $A1=0x4$ and $B1=0x3$. In this case, the trailing edge is positioned at the boundary of cycle $n+1$, which is actually considered to belong to cycle $n+2$ and therefore does not cause the output flip-flop to transition.

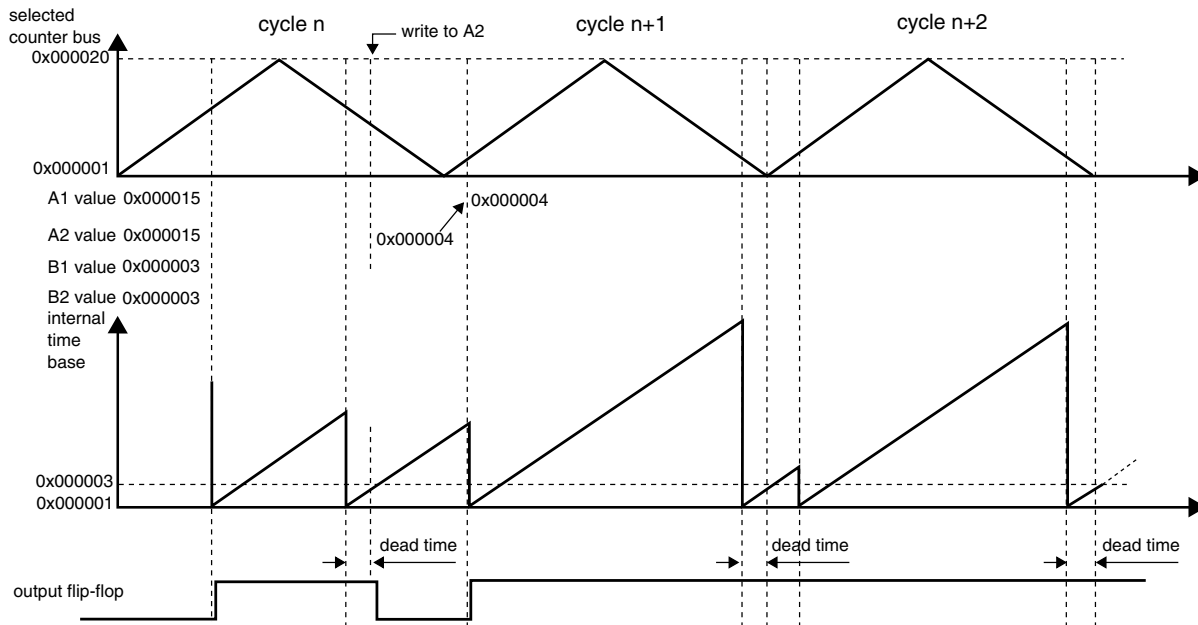


Figure 36-153. OPWMCB with 100% Duty Cycle ($A1=4$ and $B1=3$)

The output disable feature, if enabled, causes the output flip-flop to transition to the EDPOL inverted state. This feature allows an application to force the channel output pin to a "safe" state. The internal channel matches continue to occur even in this case, thus generating Flags. As soon as the output disable is deasserted, the channel output pin is again controlled by $A1$ and $B1$ matches. Note that this process is synchronous, meaning that the output channel pin transitions only on system clock edges.

It is important to notice that, such as in OPWMB and OPWFMB modes, the match signal used to set or clear the channel output flip-flop is generated on the deassertion of the channel combinational comparator output signal which compares the selected time base with $A1$ or $B1$ register values. Please refer to [Figure 36-143](#), which describes the delay from matches to output flip-flop transition in OPWFMB mode. The operation of OPWMCB mode is similar to OPWFMB regarding matches and output pin transition.

36.6.1.1.17 Output Pulse Width Modulation (OPWM) Mode

Registers $A1$ and $B1$ define the leading and trailing edges of the PWM output pulse, respectively. $MODE[6]$ bit controls the transfer from register $B2$ to $B1$, which can be done either immediately ($MODE[6]$ cleared, $MODE[0:6]=01000b0$), providing the fastest change in the duty cycle, or at every match of register $A1$ ($MODE[6]$ set, $MODE[0:6]=01000b1$).

The value loaded in register A1 is compared with the value on the selected time base. When a match on comparator A occurs, the output flip-flop is set to the value of the EDPOL bit. When a match occurs on comparator B, the output flip-flop is set to the complement of the EDPOL bit.

A FLAG can be generated at match B, when MODE[5] is cleared, or in both matches, when MODE[5] is set.

At any time, the FORCMA and FORCMB bits allow the software to force the output flip-flop to the level corresponding to a match on A or B respectively. Note that FLAG bit is not set by the FORCMA and FORCMB operations.

If subsequent matches occur on comparators A and B, the PWM pulses continue to be generated, regardless of the state of the FLAG bit.

At OPWM mode entry the output flip-flop is set to the complement of the EDPOL bit in the EMIOSC[n] register.

In order to achieve 100% duty cycle, both registers A1 and B1 must be set to the same value. When a simultaneous match on comparators A and B occur, the output flip-flop is set at every period to the value of EDPOL bit. 0% duty cycle is possible by writing 0x0 to register A. When a match occurs, the output flip-flop is set at every period to the complement of EDPOL bit. The transfer from register B2 to B1 is still controlled by the MODE[6] bit.

Note

If A1 and B1 are set to the 0x0, a 0% duty cycle waveform is produced.

[Figure 36-154](#) and [Figure 36-155](#) show the Unified Channel running in OPWM with immediate update and next period update, respectively.

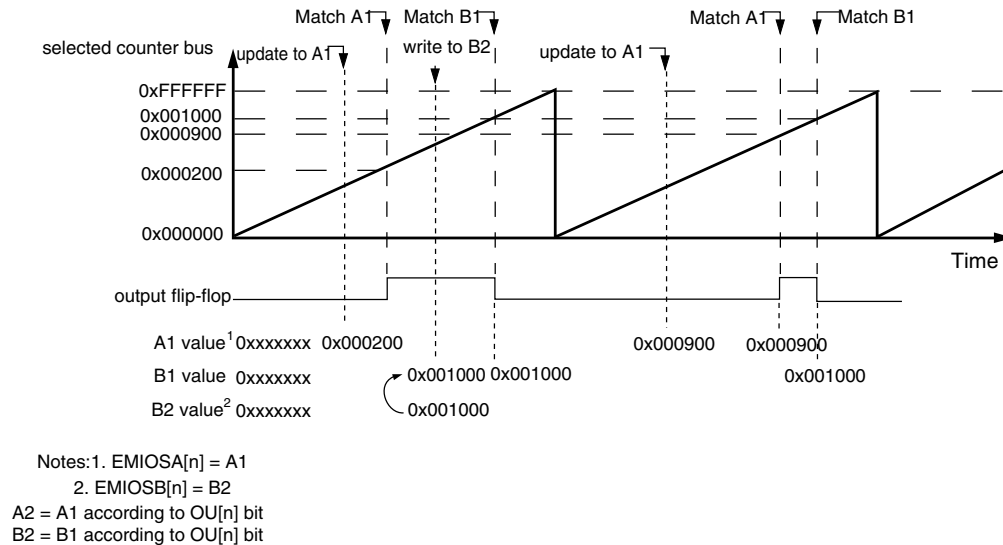


Figure 36-154. PWM with immediate update (MODE[6] is 0b)

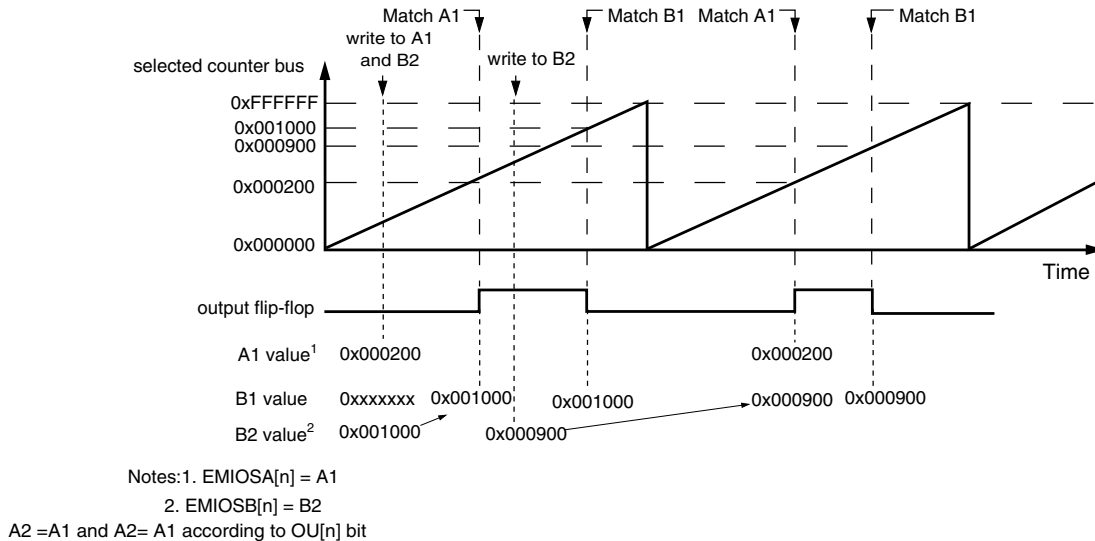


Figure 36-155. PWM with next period update (MODE[6] is 1b)

36.6.1.1.18 Output Pulse Width Modulation Buffered (OPWMB) Mode

OPWMB mode (MODE[0:6]=11000b0) is used to generate pulses with programmable leading and trailing edge placement. An external counter driven in MCB Up mode must be selected from one of the counter buses. A1 register value defines the first edge and B1 the second edge. The output signal polarity is defined by the EDPOL bit. If EDPOL is zero, a negative edge occurs when A1 matches the selected counter bus and a positive edge occurs when B1 matches the selected counter bus.

The A1 and B1 registers are double buffered and updated from A2 and B2, respectively, at the cycle boundary. The load operation is similar to the OPWFMB mode. Please refer to [Figure 36-145](#) for more information about A1 and B1 registers update.

FLAG can be generated at B1 matches, when MODE[5] is cleared, or in both A1 and B1 matches, when MODE[5] is set. If subsequent matches occur on comparators A and B, the PWM pulses continue to be generated, regardless of the state of the FLAG bit.

FORCMA and FORCMB bits allow the software to force the output flip-flop to the level corresponding to a match on A1 or B1 respectively. The FLAG bit is not set by the FORCMA and FORCMB operations.

At OPWMB mode entry the output flip-flop is set to the value of the EDPOL bit in the EMIOSC[n] register.

Some rules applicable to the OPWMB mode are:

- B1 matches have precedence over A1 matches if they occur at the same time within the same counter cycle
- A1=0 match from cycle **n** has precedence over B1 match from cycle **n-1**
- A1 matches are masked out if they occur after B1 match within the same cycle
- Any value written to A2 or B2 on cycle **n** is loaded to A1 and B1 registers at the following cycle boundary (assuming OU[n] bit of EMIOSOUDIS register is not asserted). Thus the new values will be used for A1 and B1 matches in cycle **n+1**

The following figure describes the operation of the OPWMB mode regarding A1 and B1 matches and the transition of the channel output pin. In this example EDPOL is set to zero.

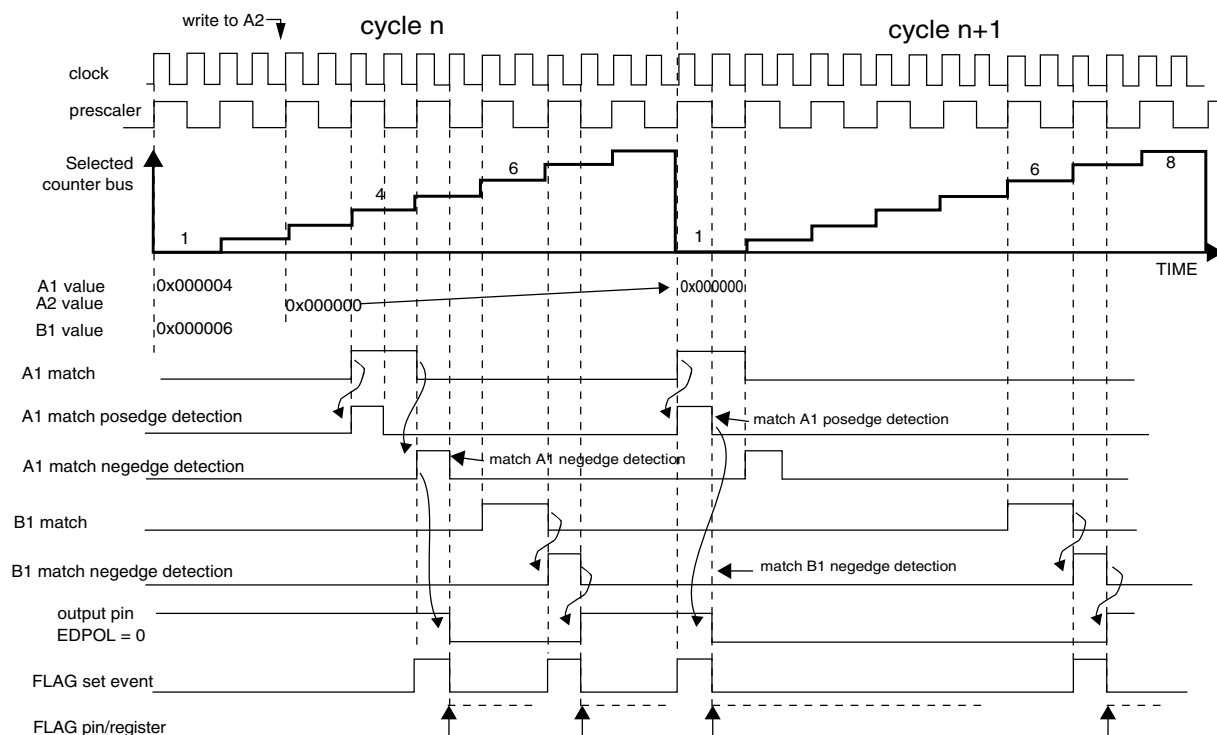


Figure 36-156. OPWMB Mode Matches and Flags

Note that the output pin transitions are based on the negedges of the A1 and B1 match signals. The figure above shows in cycle *n+1* the value of A1 register being set to zero. In this case the match posedge is used instead of the negedge to transition the output flip-flop.

The figure below describes the channel operation for 0% duty cycle. Note that the A1 match posedge signal occurs at the same time as the B1=0x8 negedge signal. In this case A1 match has precedence over B1 match, causing the output pin to remain at EDPOL bit value, thus generating a 0% duty cycle signal.

Functional Description

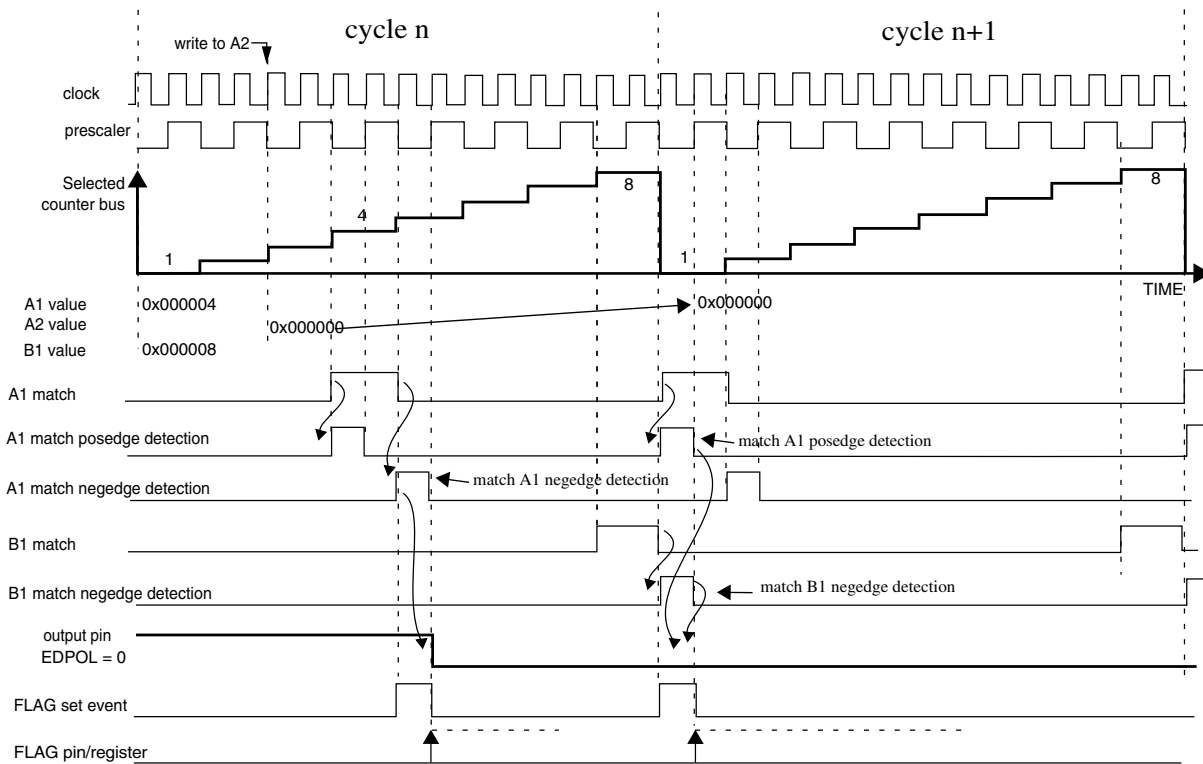


Figure 36-157. OPWMB Mode with 0% Duty Cycle

Figure 36-158 describes the operation of the OPWMB mode with the Output Disable signal being asserted. The Output Disable forces a transition in the output pin to the EDPOL bit value. After deasserted, the output disable allows the output pin to transition at the following A1 or B1 match. Note that the Output Disable does not modify the Flag bit behavior. Note that there is one system clock delay between the assertion of the output disable signal and the transition of the output pin to EDPOL.

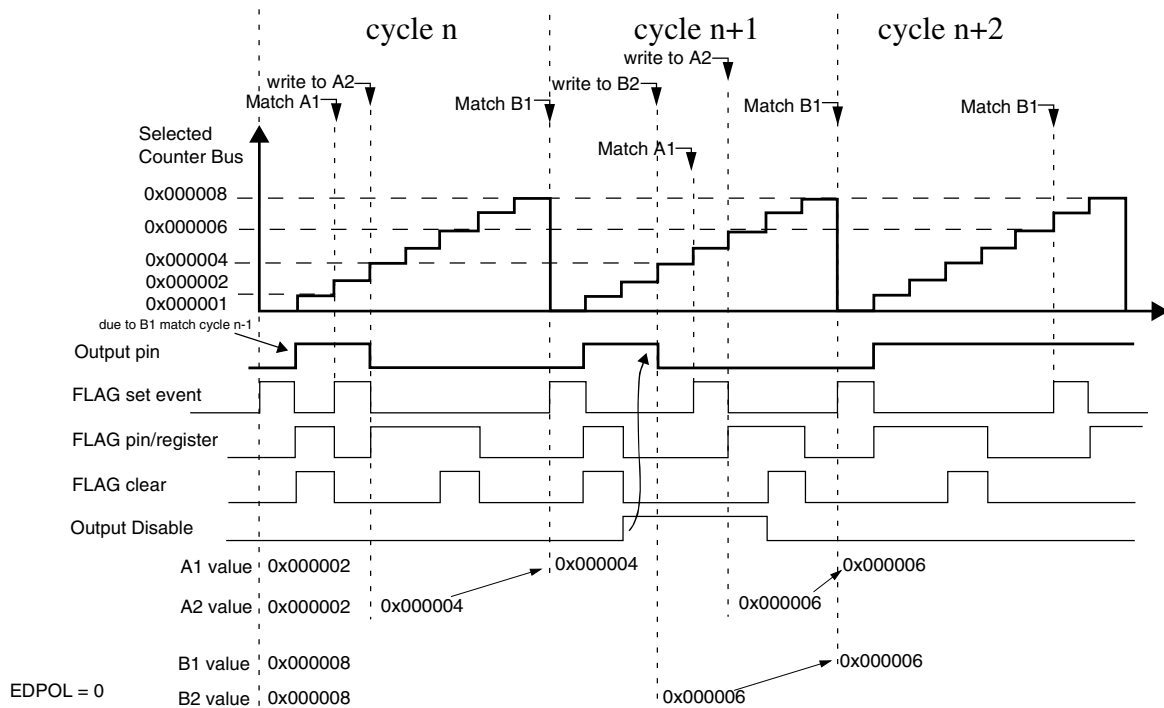


Figure 36-158. OPWMB Mode with Active Output Disable (MODE[6] bit is 1b)

Figure 36-159 shows a waveform changing from 100% to 0% duty cycle. EDPOL in this case is zero. In this example, B1 is programmed to the same value as the period of the external selected time base.

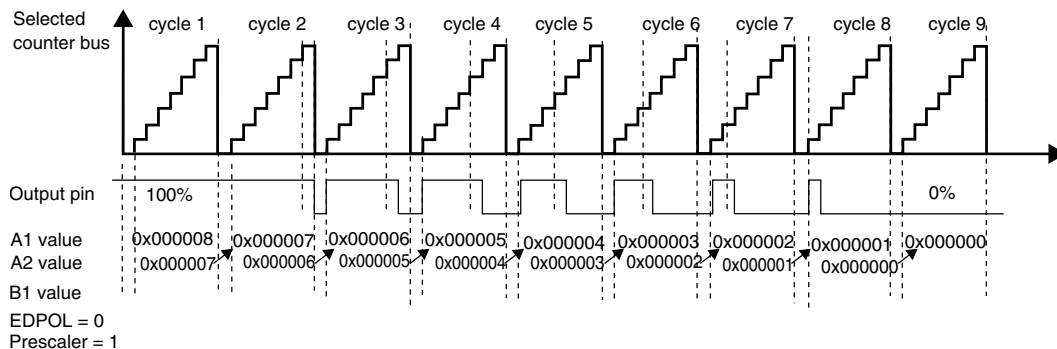


Figure 36-159. OPWMB Mode from 100% to 0% Duty Cycle

In Figure 36-159, if B1 is set to a value lower than 0x8, it is not possible to achieve 0% duty cycle by only changing A1 register value. Since B1 matches have precedence over A1 matches the output pin transitions to the opposite of EDPOL bit at B1 match. Note also that if B1 is set to 0x9, for instance, B1 match does not occur, thus a 0% duty cycle signal is generated.

36.6.1.2 Input Programmable Filter (IPF)

The IPF ensures that only valid input pin transitions are received by the Unified Channel edge detector. A block diagram of the IPF is shown in [Figure 36-160](#).

The IPF is a 5-bit programmable up counter that is incremented by the selected clock source, according to bits IF[0:3] in EMIOSC[n] register.

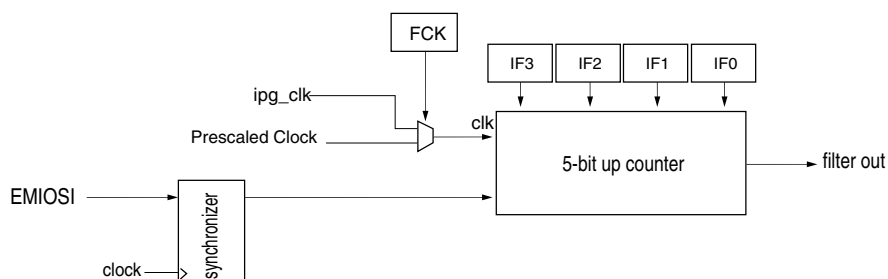


Figure 36-160. Input Programmable Filter Submodule Diagram

The input signal is synchronized by system clock. When a state change occurs in this signal, the 5-bit counter starts counting up. As long as the new state is stable on the pin, the counter remains incrementing. If a counter overflow occurs, the new pin value is validated. In this case, it is transmitted as a pulse edge to the edge detector. If the opposite edge appears on the pin before validation (overflow), the counter is reset. At the next pin transition, the counter starts counting again. Any pulse that is shorter than a full range of the masked counter is regarded as a glitch and it is not passed on to the edge detector. A timing diagram of the input filter is shown in [Figure 36-161](#).

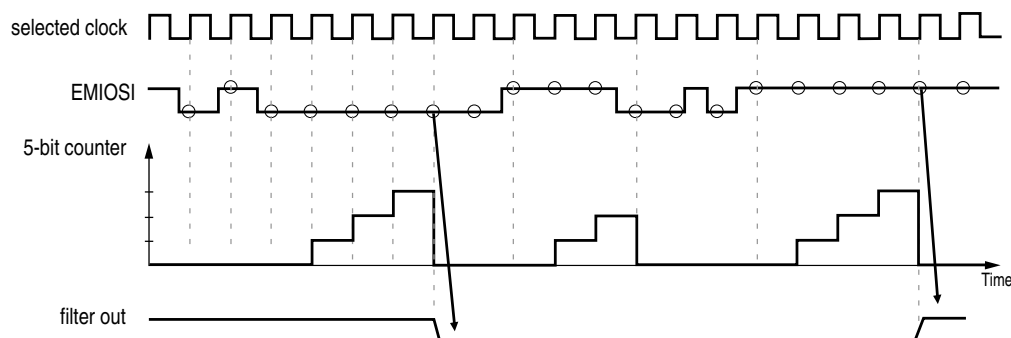


Figure 36-161. Input Programmable Filter Example (IF[0:3] is 0010b)

The filter is not disabled during either freeze state or negated GTBE input

36.6.1.3 Clock Prescaler (CP)

The CP divides the GCP output signal to generate a clock enable for the internal counter of the Unified Channels. The GCP output signal is prescaled by the value defined in the UCPRE[0:1] field description in the EMIOSC[n] register. The prescaler is enabled by setting the UCPREN bit in the EMIOSC[n] and can be stopped at any time by clearing this bit, thereby stopping the internal counter in the Unified Channel.

In order to ensure safe working and avoid glitches the following steps must be performed whenever any update in the prescaling rate is desired:

1. Write 0 at both GPREN bit in EMIOSMCR register and UCPREN bit in EMIOSC[n] register, thus disabling prescalers;
2. Write the desired value for prescaling rate at UCPRE[0:1] bits in EMIOSC[n] register;
3. Enable channel prescaler by writing 1 at UCPREN bit in EMIOSC[n] register;
4. Enable global prescaler by writing 1 at GPREN bit in EMIOSMCR register.

The prescaler is not disabled during either freeze state or negated GTBE input.

36.6.1.4 Effect of Freeze on the Unified Channel

When in debug mode, FRZ bit in the EMIOSMCR register and the FREN bit in the EMIOSC[n] are both set, the internal counter and Unified Channel capture and compare functions are halted. The UC is frozen in its current state.

During freeze, all registers are accessible. When the Unified Channel is operating in an output mode, the force match functions remain available, allowing the software to force the output to the desired level.

Note that for input modes, any input events that may occur while the channel is frozen are ignored.

When exiting debug mode or freeze enable bit is cleared (FRZ in the EMIOSMCR or FREN in the EMIOSC[n] register) the channel actions resume, but may be inconsistent until channel enters GPIO mode again.

36.6.2 IP Bus Interface Unit (BIU)

The BIU provides the interface between the Internal Interface Bus (IIB) and the Peripheral Bus, allowing communication among all submodules and this IP interface.

The BIU allows 8, 16 and 32 bits access. They are performed over a 32-bit data bus in a single cycle clock.

36.6.2.1 Effect of Freeze on the BIU

When the FRZ bit in the EMIOSMCR register is set and the module is in debug mode, the operation of BIU is not affected.

36.6.3 STAC Bus Client submodule

The STAC bus client provides one external time base, imported from the STAC bus, to the Unified Channels. [Figure 36-162](#) provides a block diagram for the module.

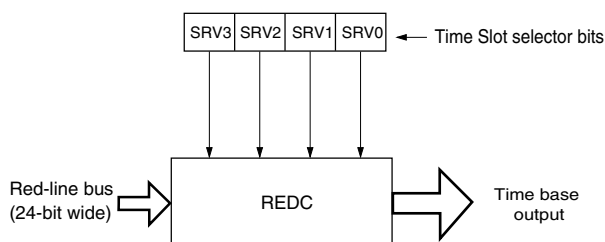
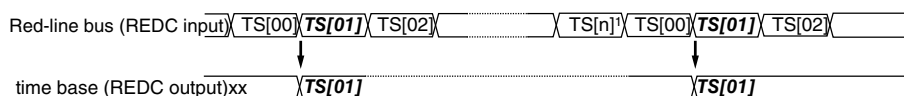


Figure 36-162. Block diagram

Bits SRV[0:3] in register EMIOSMCR selects the desired time slot of the STAC bus to be output. [Figure 36-163](#) shows a timing diagram for the REDC.



NOTES: 1. Maximum of 16 Time Slots (TS[n])
2. In this case, SRV bits were set to capture TS[01]

Figure 36-163. Timing diagram for the STAC bus and STAC output

Every time the selected time slot change, the STAC output is updated.

See the chip-specific eMIOS information for the STAC server IDs.

36.6.3.1 Effect of Freeze on the STAC bus

When the FRZ bit in the EMIOS_MCR register is set and the module is in debug mode, the operation of REDC submodule is not affected, i.e., there is no freeze function in this submodule.

36.6.4 Global Clock Prescaler Submodule (GCP)

The GCP divides the system clock to generate a clock for the CPs of the channels. The main clock signal is prescaled by the value defined in the Global Prescaler clock divider table within the eMIOS Module Configuration Register (EMIOSMCR) description according to the GPRE[0:7] bits in EMIOSMCR register. The global prescaler is enabled by setting the GPREN bit in the EMIOSMCR register and can be stopped at any time by clearing this bit, thereby stopping the internal counters in all the channels.

In order to ensure safe working and avoid glitches the following steps must be performed whenever any update in the prescaling rate is desired:

1. Write 0 at GPREN bit in EMIOSMCR register, thus disabling global prescaler;
2. Write the desired value for prescaling rate at GPRE[0:7] bits in EMIOSMCR register;
3. Enable global prescaler by writing 1 at GPREN bit in EMIOSMCR register.

The prescaler is not disabled during either freeze state or negated GTBE input.

36.6.4.1 Effect of Freeze on the GCP

When the FRZ bit in the EMIOSMCR register is set and the module is in debug mode, the operation of GCP submodule is not affected, i.e., there is no freeze function in this submodule.

36.7 Initialization/Application Information

On resetting the eMIOS, the Unified Channels enter GPIO input mode.

36.7.1 Considerations

Before changing an operating mode, the UC must be programmed to GPIO mode and EMIOSA[n] and EMIOSB[n] registers must be updated with the correct values for the next operating mode. Then the EMIOSC[n] register can be written with the new operating mode. If a UC is changed from one mode to another without performing this procedure, the first operation cycle of the selected time base can be random, i.e., matches

can occur in random time if the contents of EMIOSA[n] or EMIOSB[n] were not updated with the correct value before the time base matches the previous contents of EMIOSA[n] or EMIOSB[n].

When interrupts are enabled, the software must clear the FLAG bits before exiting the interrupt service routine.

36.7.2 Application Information

Correlated output signals can be generated by all output operation modes. Bits OU[n] of eMIOS_OUDIS register can be used to control the update of these output signals.

In order to guarantee that the internal counters of correlated channels are incremented in the same clock cycle, the internal prescalers must be set up before enabling the global prescaler. If the internal prescalers are set after enabling the global prescaler, the internal counters may increment in the same ratio, but at a different clock cycle.

It is recommended to drive Output Disable Input signals with the emios_flag_out signals of some UCs running in SAIC mode. When an output disable condition happens, the software interrupt routine must service the output channels before servicing the channels running SAIC. This procedure avoid glitches in the output pins.

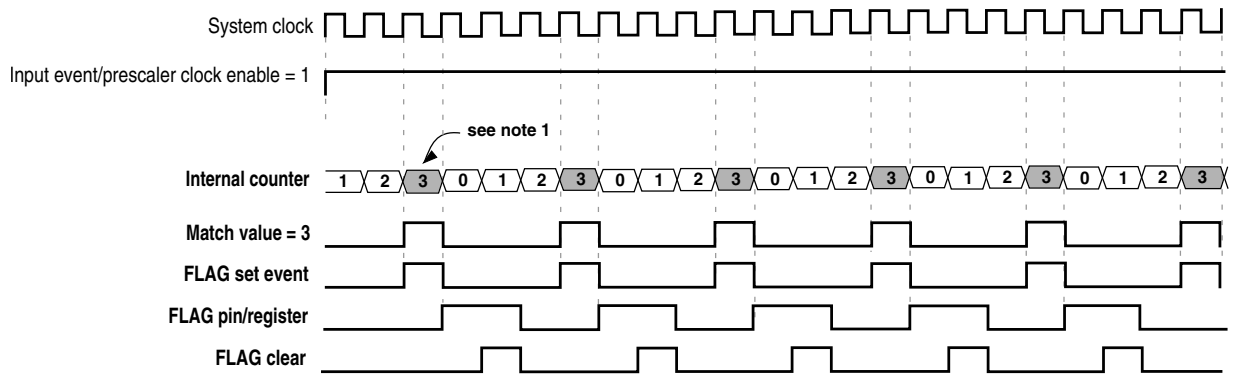
36.7.2.1 Time Base Generation

For MC and OPWFM with internal clock source operation modes, the internal counter rate can be modified by configuring the clock prescaler ratio. [Figure 36-164](#) shows an example of a time base with prescaler ratio equal to one.

Note

MCB and OPWFMB modes have a different behavior.

PRE SCALED CLOCK RATIO = 1 (bypassed)

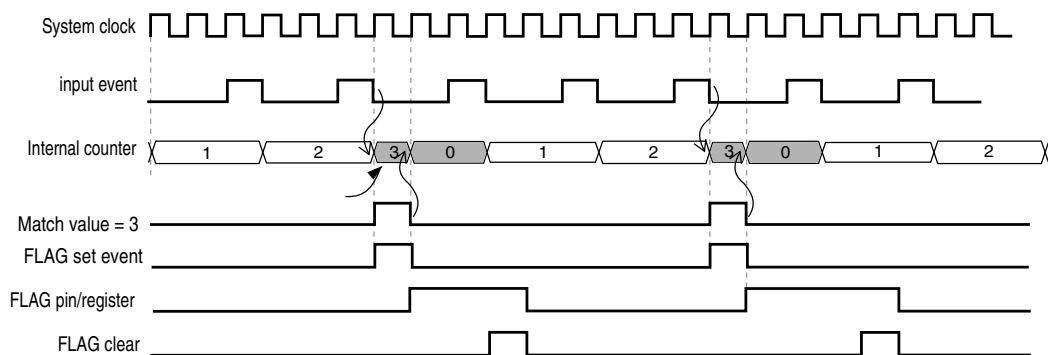
**Figure 36-164. Time base period when running in the fastest prescaler ratio**

If the prescaler ratio is greater than one or external clock is selected, the counter may behave in one of four different ways depending on the channel mode:

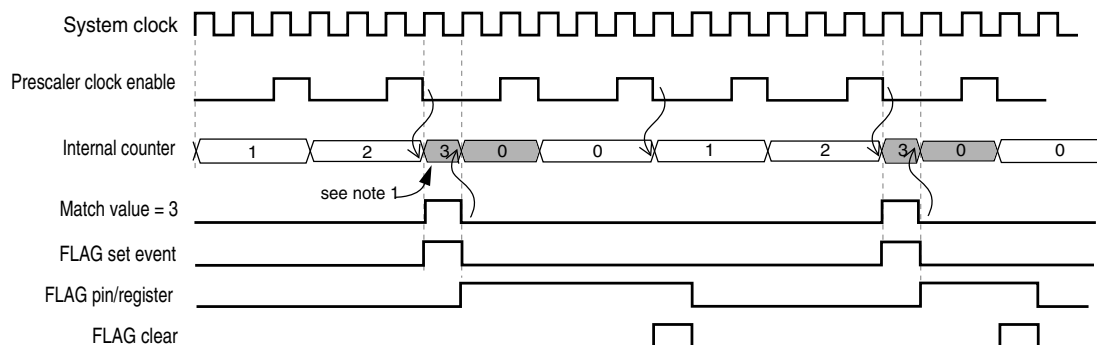
- If MC mode and Clear on Match Start and External Clock source are selected the internal counter behaves as described in [Figure 36-165](#).
- If MC mode and Clear on Match Start and Internal Clock source are selected the internal counter behaves as described in [Figure 36-166](#).
- If MC mode and Clear on Match End are selected the internal counter behaves as described in [Figure 36-167](#).
- If OPWFM mode is selected the internal counter behaves as described in [Figure 36-166](#). The internal counter clears at the start of the match signal, skips the next prescaled clock edge and then increments in the subsequent prescaled clock edge.

Note

MCB and OPWFMB modes have a different behavior.

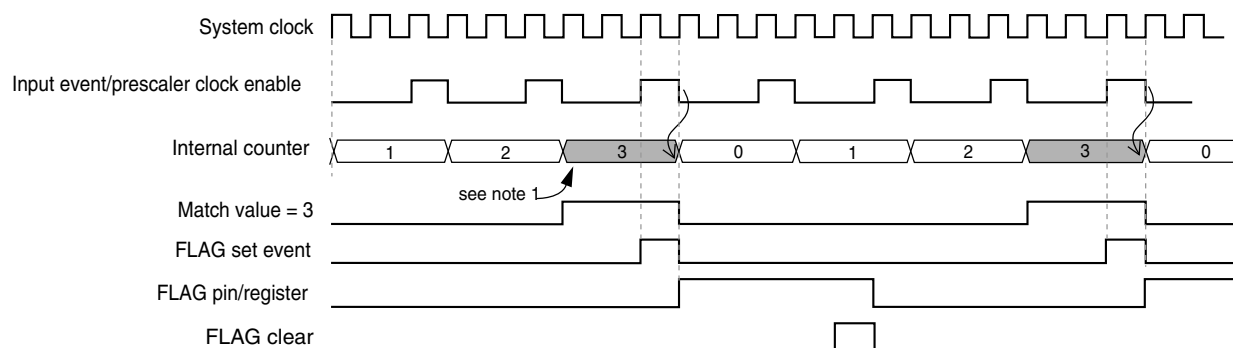
**Figure 36-165. Time base generation with external clock and clear on match start**

PRESCALED CLOCK RATIO = 3



Note 1: When a match occurs, the first clock cycle is used to clear the internal counter, and only after a second edge of pre scaled clock the counter will start counting.

Figure 36-166. Time base generation with internal clock and clear on match start



Note 1: The match occurs only when the input event/prescaler clock enable is active. Then, the internal counter is immediately cleared.

Figure 36-167. Time base generation with clear on match end

36.7.2.2 Coherent Accesses

Reading the EMIOSA[n] register again in the same period of the last read of EMIOSB[n] register may lead to incoherent results. This will occur if the last read of EMIOSB[n] register occurred after a disabled B2 to B1 transfer.

36.7.2.3 Channel/Modes Initialization

The following basic steps summarize basic output mode startup, assuming the channels are initially in GPIO mode:

1. *[global]* Disable Global Prescaler.
2. *[timebase channel]* Disable Channel Prescaler.

3. [*timebase channel*] Write initial value at internal counter.
4. [*timebase channel*] Set A/B register.
5. [*timebase channel*] Set channel to MC(B) Up mode.
6. [*timebase channel*] Set prescaler ratio.
7. [*timebase channel*] Enable Channel Prescaler.
8. [*output channel*] Disable Channel Prescaler.
9. [*output channel*] Set A/B register.
10. [*output channel*] Select timebase input through BSL[1:0] bits.
11. [*output channel*] Enter output mode.
12. [*output channel*] Set prescaler ratio (same ratio as timebase channel).
13. [*output channel*] Enable Channel Prescaler.
14. [*global*] Enable Global Prescaler.
15. [*global*] Enable Global Time Base.

The timebase channel and the output channel may be the same for some applications such as in OPWFM(B) mode or whenever the output channel is intended to run the timebase itself.

At any time the flags can be configured.

Chapter 37

Enhanced Time Processing Unit (eTPU)

37.1 Chip-specific eTPU information

This chip contains three eTPU modules with the following configuration:

- 32 channels per eTPU module
- 36 KB code RAM total
- 9 KB parameter RAM total
- ETPU_A and ETPU_B share the same program and data (parameter RAM) space in a dual-engine configuration
 - Support for up to 200 MHz clock rate (provides 100 MHz instruction execution rate)
 - ECC support on parameter RAM and program space
 - Sizes of shared eTPU_A and eTPU_B memories:
 - SCM instruction memory address space: 24 KB code RAM
 - SDM data memory address space: 6 KB parameter RAM
 - No shared subsystem support and therefore no Shared Subsystem Integration registers (SSIRs)
- ETPU_C has private program and parameter RAM space in a single-engine configuration
 - Support for up to 200 MHz clock rate (provides 100 MHz instruction execution rate)
 - ECC support on parameter RAM and program space
 - Sizes of eTPU_C memories:
 - SCM instruction memory address space: 12 KB code RAM
 - SDM data memory address space: 3 KB parameter RAM
 - Supports REACM2 by [Shared Subsystems support](#)

See also this information:

- [eMIOS global time base synchronization with eTPU](#)

- [DSPI connections with SIU, eTPU, and eMIOS modules](#)
- [eMIOS and eTPU ODIS connections](#)

37.1.1 eTPU register field reset values

The following table lists chip-specific reset values for register fields of the eTPU instances.

Table 37-1. Chip-specific eTPU register field reset values

eTPU instance	Register field	Reset value	Notes
eTPU_A/B	eTPU_SCR_MCR[SCMSIZE]	01011b	The reset value of eTPU_SCR_MCR is 000B_0000h.
eTPU_A/B	eTPU_TBR_STAC_ENG1[SERVER_ID1]	0000b	This is consistent with eTPU_A TCR1 in Table 37-2 .
eTPU_A/B	eTPU_TBR_STAC_ENG1[SERVER_ID2]	0010b	This is consistent with eTPU_A TCR2 in Table 37-2 .
eTPU_A/B	eTPU_TBR_STAC_ENG2[SERVER_ID1]	0001b	This is consistent with eTPU_B TCR1 in Table 37-2 .
eTPU_A/B	eTPU_TBR_STAC_ENG2[SERVER_ID2]	0011b	This is consistent with eTPU_B TCR2 in Table 37-2 .
eTPU_C	eTPU_SCR_MCR[SCMSIZE]	00101b	The reset value of eTPU_SCR_MCR is 0005_0000h.
eTPU_C	eTPU_TBR_STAC_ENG1[SERVER_ID1]	0100b	This is consistent with eTPU_C TCR1 in Table 37-2 .
eTPU_C	eTPU_TBR_STAC_ENG1[SERVER_ID2]	0101b	This is consistent with eTPU_C TCR2 in Table 37-2 .
eTPU_C	eTPU_TBR_STAC_ENG2[SERVER_ID1]	—	eTPU_C does not have engine 2.
eTPU_C	eTPU_TBR_STAC_ENG2[SERVER_ID2]	—	eTPU_C does not have engine 2.

37.1.2 Access to eTPU registers in chip-level Stop mode

On this chip, if you attempt to access eTPU registers when the chip is in Stop mode (as defined in [Modes of operation](#)), you will receive a bus error.

NOTE

This Stop mode corresponds to the chip-level Stop mode. It is different from the eTPU Module Disable mode, also defined in [Modes of operation](#).

37.1.3 STAC server IDs

On this chip, an eTPU module can export the TCR1 and TCR2 timebases on the STAC bus to an eMIOS or eQADC module. See [STAC interface](#) for more information.

An eTPU module is able to export a timebase to an eMIOS or eQADC module when the eTPU timebase period is equal to or greater than 6 times the eTPU clock frequency, regardless of whether the clock ratio between the eTPU and eMIOS or eQADC is 1:1 or 2:1.

STAC server IDs are defined in the following table.

No more than 4 STAC servers must be enabled at any time.

Table 37-2. STAC server IDs

eTPU instance	Time base	Server ID
eTPU_A	TCR1	0
eTPU_A	TCR2	2
eTPU_B	TCR1	1
eTPU_B	TCR2	3
eTPU_C	TCR1	4
eTPU_C	TCR2	5

37.1.4 eTPU inputs with Glitch Filters

See [IGF interaction with other modules](#) for connections between eTPU and Input Glitch Filters (IGFs).

37.1.5 eMIOS and eTPU signal passthrough from pins via IGFs

On this chip, the IGF modules mediate between some pins and the eMIOS and eTPU modules. After system reset, the IGF modules are in Configuration mode by default. This mode disables IGF output and prevents the signals from the pins from reaching the eMIOS and eTPU modules.

The signals affected by this situation appear in [IGF interaction with other modules](#).

To route an affected signal directly to an eMIOS or eTPU instance, program the applicable channel of the applicable IGF instance:

1. Ensure that IGF_MCRn[MDIS] is 0 and that the IGF channel is disabled:
IGF_MCRn[FGEN] is 0.
2. Configure the rising and falling edges for bypass:

- Rising edge: Write 00b to IGF_MCRn[RFM].
- Falling edge: Write 00b to IGF_MCRn[FFM].

3. Enable the IGF channel: Write 1 to IGF_MCRn[FGEN].

At the end of this sequence, the programmed filter enters Normal mode and passes through the affected signal.

37.1.6 eTPU output disable channel groups

The output disable channel groups are defined in the following table. See [Table 36-2](#) for additional information.

Table 37-3. Output disable channel groups

Output disable signal	eTPU_A channels	eTPU_B channels	eTPU_C channels
ODIS_1(0)	0 to 7	–	0 to 7
ODIS_1(1)	8 to 15	–	8 to 15
ODIS_1(2)	16 to 23	–	16 to 23
ODIS_1(3)	24 to 31	–	24 to 31
ODIS_2(0)	–	0 to 7	–
ODIS_2(1)	–	8 to 15	–
ODIS_2(2)	–	16 to 23	–
ODIS_2(3)	–	24 to 31	–

37.1.7 Control of the eTPU Output Buffer Enable signal

On this chip, the eTPU Output Buffer Enable signal is controlled by the Pad Configuration Registers (PCRs) in the SIU. See [Table 8-3](#) and [Pad Configuration Register \(SIU_PCRn\)](#).

37.2 Device-specific Information

- Single (32 channels) or Dual (64 channels, dual core) Engine
- SCM size, error detection, error correction
- SDM size, error detection, error correction

- Nexus class 3 support
- GTBE is a logical OR with other modules (eMIOS, other eTPU)
- channel output buffers are controlled through the SIU, and cannot be controlled with microcode
- channels used to trigger the eQADC
- channels used to enable the Reaction Module modulation
- channels with DMA request
- TRC1, TRC2 STAC server IDs

37.3 Introduction

eTPU is an intelligent, semi-autonomous co-processor designed for timing control. Operating in parallel with the Host CPU, the eTPU processes instructions and real-time input events, performs output waveform generation, and accesses shared data without Host intervention. Consequently, for each timer event, the Host CPU setup and service times are minimized or eliminated.

High-level assembler, compiler and documentation allows customers to develop their own functions on the eTPU.

eTPU architecture aims at high resolution timing capabilities. From a system perspective, high resolution timing is limited by Host CPU overhead required for servicing timing tasks such as period measurement, pulse measurement, pulse width modulated waveform generation, etc. On the eTPU, high resolution timing is achieved by three main capabilities:

- Reduced latency: pin actions are immediate.
- Reduce or eliminate host interrupt service time.
- Double action channel capability reducing the channel request rate.

eTPU provides higher resolution than the Host CPU can achieve and creates no Host overhead for servicing timing tasks.

Latency is the interval from occurrence of an event to the start of event servicing. eTPU can service its own events without interrupting the Host. There are two types of timing events:

- Input pin transition.
- Selected Time Base match, i.e., a selected Time Base counter reached or exceeded a pre-programmed value.

Service time is the time spent servicing an event. In general, in microcontrollers the service time is constrained because the instruction set is not optimized for time function synthesis. The eTPU instruction set is optimized, so that time functions can be implemented with much fewer instructions than the Host CPU. Instructions execute faster, service time is reduced and program memory compacted.

Instructions executed by the eTPU are connected directly to the eTPU timing hardware and allow parallelism of hardware related actions.

37.3.1 Overview

The figure below shows a top-level eTPU block diagram. It displays a dual eTPU Engine configuration.

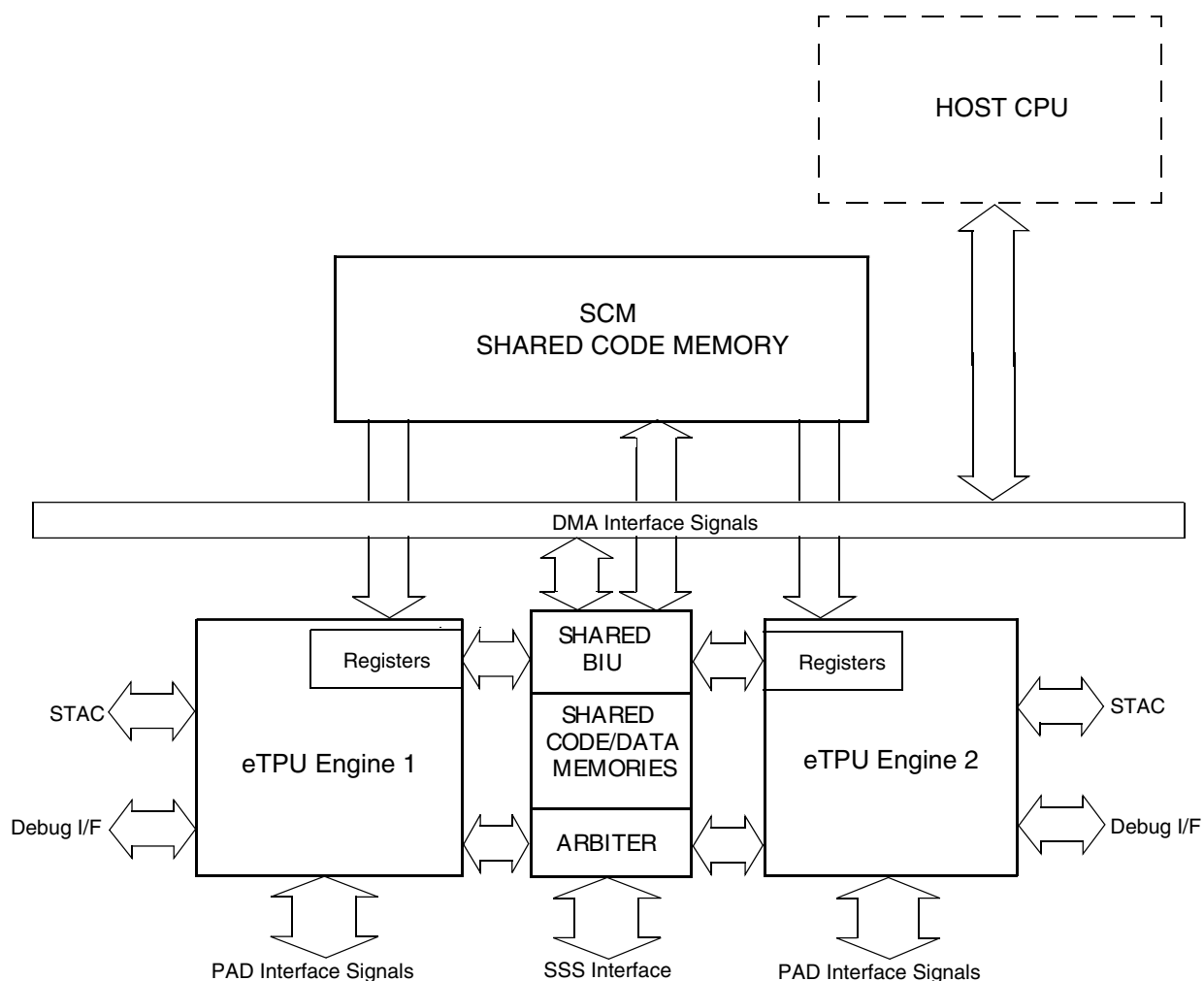


Figure 37-1. eTPU Block Diagram

eTPU Engine is responsible for processing input pin transitions and output pin waveform generation based on the **Time Bases**. Each eTPU Engine has its own microprocessor and dedicated hardware for processing signals on I/O pins and can also interface with external time bases through the STAC bus.

Both eTPU Engine CPUs, hereafter called **microengines**, fetch microinstructions from a **Shared Code Memory - SCM**.

Shared Data Memory - SDM - holds eTPU application parameters and work data. It is accessed by Host and both microengines.

Bus Interface Unit - BIU - allows Host to access eTPU registers, SCM and SDM.

Each I/O signal pair is associated with a dedicated **Channel**, which provides hardware for input signal processing and output signal generation, in relationship with selected Time Bases.

The eTPU, as a microprocessed subsystem, works much like a typical real-time system: it runs microengine code from instruction memory (SCM) to handle specific events, accessing data memory (SDM) for parameters, work data and application state info; events may originate from I/O Channels (due to pin transitions and/or time base matches), Host CPU requests or inter-channel requests; events that call for local eTPU processing activate the microengine by issuing a **Service Request**. The Service Request microcode may set an interrupt to the Host CPU. I/O channel events cannot directly interrupt the Host CPU.

Each Channel is associated with a **Function**, which defines its behavior: the Function is a software entity consisting, within the eTPU, of a set of microengine routines that attend to Service Requests. The Function routines are also responsible for Channel configuration. Function routines reside in SCM, which may contain several Functions. A Function may be assigned to several Channels, but a Channel can be associated with just one Function at a given moment. The association between Functions and Channels is defined by Host CPU, and is explained in detail in [Functions and threads](#).

eTPU hardware supplies resource sharing features that support concurrency:

- A hardware **Scheduler** dispatches the Service Request microengine routines based on a set of priorities defined by the Host CPU. Each Channel has its associated priority.
- A Service Request routine cannot be interrupted until it ends. This sequence of uninterrupted instruction execution is called a **Thread**.
- Channel-specific context (registers and flags) is automatically switched between the end of a Thread and the beginning of the next one.
- SDM arbitration, a dual-parameter coherency controller and semaphores can be used to ensure coherent access to eTPU data shared by both eTPU Engines and Host CPU.

37.3.1.1 eTPU Engine

Each eTPU Engine consists of all blocks that are duplicated in a dual eTPU configuration: two 24-bit time bases, 32 independent timer channels, a task scheduler, a microengine, and a Host interface. In addition, a 32-bit Shared Data Memory (SDM) is used for two eTPU Engines data storage and for passing information between the eTPU Engines and the Host CPU.

The following figure shows the block diagram for the eTPU Engine.

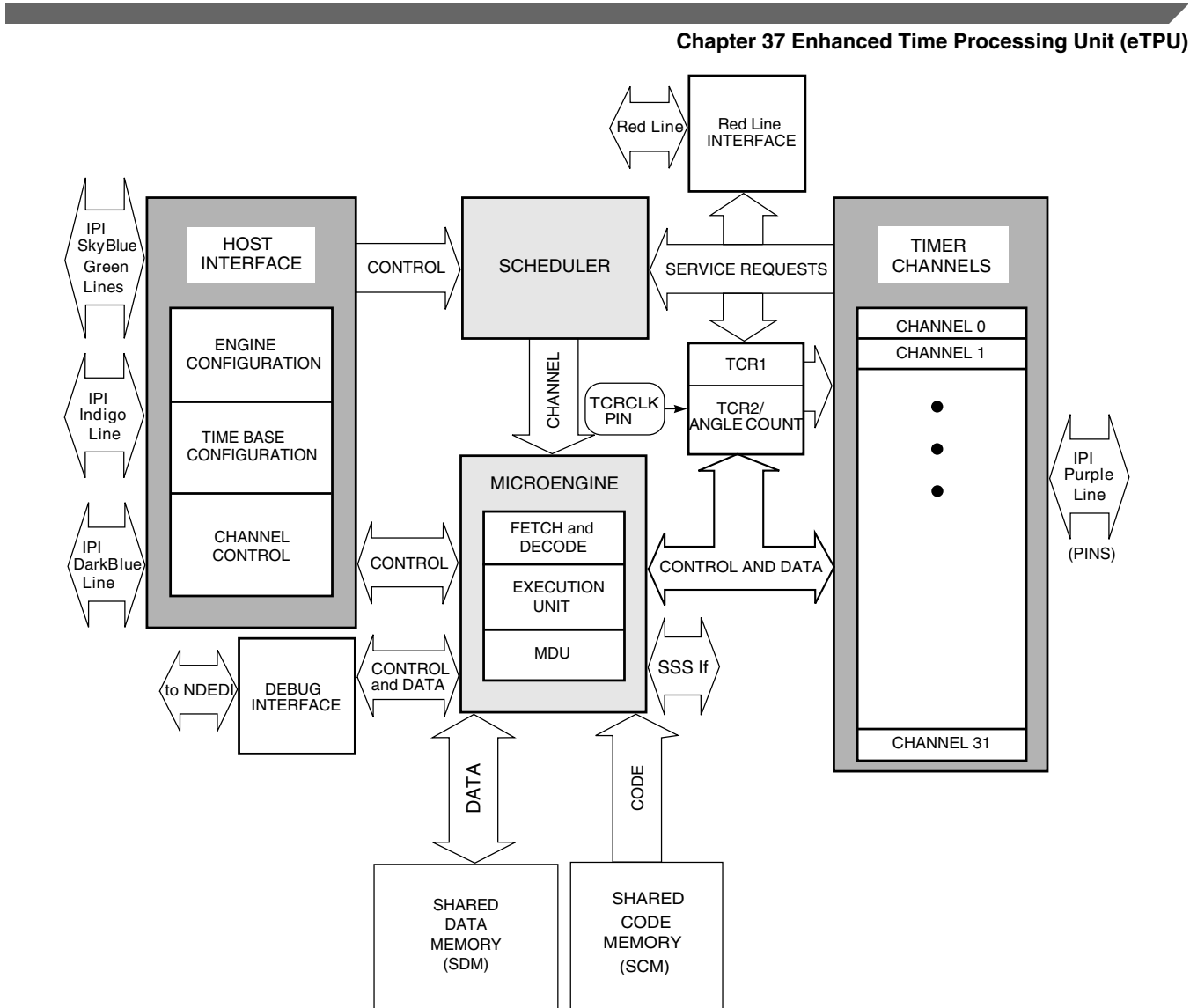


Figure 37-2. eTPU Engine Block Diagram

eTPU Engines 1 and 2 are sometimes called eTPU 1 and eTPU 2 for short throughout this chapter.

37.3.1.1.1 Time Bases

Two 24-bit counters TCR1 and TCR2 provide reference time bases for all match and input capture events. Prescalers for both time bases are controlled by the Host CPU through bit fields in the eTPU Engine configuration registers. The eTPU is able to export/import time to/from TCR1 or TCR2 in accordance to the STAC bus specification.

The clock for each of TCR1 and TCR2 clock can be independently derived from the eTPU clock or from an external input via the TCRCLK clock pin. In addition, the TCR2 timebase can be derived from special angle-clock hardware which enables implementing angle-based functions. This feature is added to support advanced angle based engine control applications.

For further details refer to [Time bases](#).

37.3.1.1.2 eTPU Timer Channels

Each eTPU Engine has 32 independent channels, each corresponding to an Input/Output signal pair. The channel time resolution is 24 bits, and are all identical.

Each channel consists of logic which supports two events and output controls. The event logic contains two 24-bit capture registers, two 24-bit match registers, greater-equal and equal-only comparators. Supporting two events enables many combinations of double action functions (for example the channel can handle two events with a single microcode service).

The channel configuration can be changed immediately by the microengine. Each channel can perform double capture, double match and other capture-match combinations. Channel modes available can do ordered or unordered match. Some modes are also provided that can block one match by the occurrence of the other. Service request can be generated on one or both of the match events.

Input signal can be separated from output signal in each channel. They can, optionally, be combined in a single I/O pin driver. An output buffer enable is controlled by the SIU (or SIUL). Digital filters are provided for the input signals, with distinct filtering modes available.

Each channel can use any time base or angle counter for either match or capture operation. For example, a match on TCR1 can capture the value of TCR2. The channels can request service from the microengine due to recognized pin transitions (input events) or timebase matches.

The eTPU channels also support the basic single-action operations found on TPU3 functionality with the exception that time resolution is 24-bits.

Channel configuration combinations:

- Single input capture, no match (TPU3 functionality)
- Single input capture with single match timeout (TPU3 functionality)
- Single input capture with double match timeout with several double match sub-modes
- Double input capture with single or double match timeout with several double match sub-modes
- Single output match (TPU3 functionality)

- Double output match with several double match sub-modes
- Input-dependent output generation

The double match functionality has various combinations for generation of service request and determining pin actions. For more details refer to [Enhanced Channels](#).

In addition to the predefined channel configurations above, the user can also program its own channel configuration, defining how input captures, matches and service-requests are related.

37.3.1.1.3 Host interface

The Host interface allows the Host CPU to control the operation of the eTPU. The Host CPU must initialize the eTPU by writing to the appropriate Host interface registers to assign a Function and priority to each channel. In addition, the Host writes to the Host Service Request and channel configuration registers to further define Function operation for each initialized channel. Refer to [Host interface](#) for a detailed description.

When the SCM is implemented by RAM, the Host must first initialize it with the proper microcode program prior to enabling any eTPU Function, and then enable eTPU access (which also disables Host access).

37.3.1.1.4 Shared Data Memory - SDM

The SDM works as data RAM which can be accessed by the Host CPU and up to two eTPU Engines. This memory is used for information transfer between the Host CPU and the eTPU, as data storage for the eTPU microcode program or for communication between the two eTPU Engines. SDM width is 32 bits, and is accessible by the Host as byte, 16-bit or 32-bit wide. eTPU can access it as full 32-bits, lower 24-bits or upper byte (8-bit).

The host can also access the SDM space mirrored in other area with Parameter Sign Extension (PSE). Parameter Sign Extension accesses differ from the usual host accesses to the original SDM area as follows:

- Writes are effective only to the lower 3 bytes of a word: the word's most significant byte is kept unaltered in SDM.
- Reads return the lower 3 bytes of a word sign-extended to 32 bits, i.e.: the most significant bit of the word's 2nd most significant byte is copied in all 8 bits of the most significant read byte.

Each eTPU channel can be associated with a variable number of parameters located in the SDM, according to its selected Function. In addition, the SDM can be fully shared between two eTPU Engines, enabling direct communication between them.

High flexibility of the SDM utilization is achieved as follows:

- Each channel has a programmable base address pointing to the address of its first parameter with two parameter granularity. This way the SDM can be partitioned according to the actual function needs.
- The microcode can access the first 128 parameters of the selected channel in channel relative access mode.
- Each Engine can access all the SDM address space in indirect addressing mode. Blocks of data are easily transferred using stack operation.
- Absolute addressing mode can access the first 256 parameters (TPU3 functionality), implementing a shared pool of parameters holding global variables.

In the Host address space each parameter occupies four bytes. eTPU usage of the upper byte is achieved by having a 32-bit P register which can access the upper byte, the lower 24 bits or all the 32 bits. The microcode can switch between access sizes at any time.

Each Function may require a different number of parameters. During the eTPU initialization the Host has to program channel base addresses, allocating proper parameters for each channel according to its selected Function.

37.3.1.1.5 Shared Subsystem Area

The Shared Subsystem (SSS) area is a memory map region used by both the host and the eTPU to access registers of the reaction channel, giving the user the option to control them by the host, the eTPU code, or a combination of both.

37.3.1.1.6 Scheduler

Out of reset, all channels are disabled. The Host CPU makes a channel active by assigning it one of three priorities: high, middle, or low. The Scheduler determines the order in which channels are serviced based on channel number and assigned priority. The priority mechanism, implemented in hardware, ensures that all requesting channels are serviced. For additional details refer to [Scheduler](#).

37.3.1.1.7 Microengine

eTPU microengine is a simple RISC implementation which performs each instruction in a microcycle of two eTPU clocks, while prefetching the next instruction through an instruction pipeline. Instruction execution time is constant unless it gets wait states from the SDM arbitration. Two eTPU Engines share code memory without having any performance degradation by interleaving their accesses (the Shared Code Memory has one-clock access time).

Instruction width is 32 bits. The microengine instruction set provides basic arithmetic and logic operations, flow control (jumps and subroutine calls), SDM access, and Channel configuration and control. The instruction formats are defined in such a way that allow particular combinations of two or three of these operations with unconflicting resources to be executed in parallel in the same microcycle.

Microengine has also an independent Multiply/Divide/MAC unit that performs these complex operations in parallel with other microengine instructions.

Channel functionality is tightly integrated to the instruction set through Channel Control operations and conditional Branch operations, which support jumps/calls on Channel-specific conditions. This allows quick and terse Channel configuration and control code, contributing to reduced service time.

Detailed description can be found in [Microengine](#).

37.3.1.1.8 Dual eTPU Engine System

A typical eTPU implementation includes two eTPU Engines sharing SDM and the same code in SCM.

The two eTPU Engines share the Bus Interface Unit (BIU) and the data memory (SDM) which enable Host-eTPU and eTPU Engine-Engine communication. The shared BIU includes coherency logic which supports dual parameter (8 bytes) coherency in transfers between Host and eTPU, using a temporary parameter area within the SDM. More details on this can be found on [Parameter sharing and coherency](#).

37.3.2 Features

The following sections show eTPU features and enhancements.

37.3.2.1 eTPU Feature Summary

The eTPU includes these distinctive features:

- up to 32 channels for each eTPU Engine, each channel is associated with an Input/Output signal pair.
 - enhanced input digital filters on the input pins for improved noise immunity. The eTPU digital filter can use 2 samples, 3 samples or work in continuous mode.
 - identical, orthogonal channels, except for channel 0: each channel can perform any time function. Each time function can be assigned to more than one channel at a given time, so each signal can have any functionality. Channel 0 has the same capabilities of the others, but can also work with special Angle Counter logic (see below).
 - Link Service Request allows activation of a Channel function by request of another channel, even between eTPU Engines.
 - Host Service Request allows activation of a Channel function by Host CPU request
 - each channel has an event mechanism which supports single and double action functionality in various combinations. It includes two 24-bit capture registers, two 24-bit match registers, 24-bit greater-equal and equal-only comparators.
- Two independent 24-bit time bases for channel synchronization:
 - first time base clocked by eTPU clock with programmable prescaler division from 1 to 512 (in steps of 2), or by output of second time base prescaler.
 - first time base can also be clocked by external signal with programmable prescaler division of 1 to 256.
 - second time base clocked by external signal with programmable prescaler division from 1 to 64.
 - second time base external clock source can be replaced by eTPU clock divided by 8.
 - both time bases can be exported or imported via Shared Time and Counter bus.
 - second time base counter can work as an Angle counter, enabling angle based applications to match angle instead of time.
 - second time base can also be used as a pulse accumulator gated by external signal.
- Event-Triggered RISC processor (microengine):
 - 3 stage pipeline implementation (fetch, decoding and execution), with separate instruction memory - SCM - and data memory - SDM (Harvard architecture)

- fixed-length instruction execution in one microcycle of two eTPU clocks
- interleaved SCM access in dual eTPU Engine avoids contention in time for instruction memory
- SCM address space of up to 16K positions (64 Kbytes) (see chip-specific eTPU information)
- SDM with interleaved access in dual eTPU Engine avoids contention for data memory
- SDM address space (both Engines) of up to 8 Kbytes (see chip-specific eTPU information)
- instruction set with embedded Channel support, including specialized Channel control subinstructions and conditional branching on Channel-specific flags.
- channel-oriented addressing: channel-bound address mode with Host configured Channel Base Address allows channel data isolation, independent of microengine application code.
- channel-bound data address space of up to 128 32-bit parameters (512 bytes)
- global parameter address mode allows access to common Channel data of up to 256 32-bit parameters (1024 bytes)
- support for indirect and stacked data access schemes.
- parallel execution of: data access, ALU, Channel control and flow control subinstructions in selected combinations.
- 32-bit microengine registers and 24-bit resolution ALU, with 1 microcycle addition and subtraction, absolute value, bitwise logical operations on 24-bit, 16-bit, or byte operands; single bit manipulation, shift and rotate operations, sign extension and conditional execution.
- additional 24-bit Multiply/MAC/Divide unit which supports all signed/unsigned Multiply/MAC combinations, and unsigned 24-bit Divide. The MAC/Divide unit works in parallel with the regular microcode commands.
- Resource sharing features support channel sharing of channel registers, memory and microengine time:
 - hardware Scheduler works as a "task management" unit, dispatching event service routines by predefined, Host-configured priority.

- automatic Channel context switch when a "task switch" occurs, i.e., one Function Thread ends and another begins to service a request from other Channel: Channel-specific registers, flags and parameter base address are automatically loaded for the next serviced channel.
- individual channel priority setting in 3 levels: high, middle and low.
- Scheduler priority scheme allows calculation of worst case latency for event servicing and ensures servicing all channels by preventing permanent blockage.
- SDM shared between Host CPU and both eTPU Engines, supporting communication either between Channels and Host or inter-channel.
- hardware implementation of 4 Semaphores supports resource sharing between both eTPU Engines.
- Hardware semaphores directly supported by the microengine instruction set.
- dual parameter coherency hardware support allows atomic (to host) access to 2 parameters by microengine(s) in back-to-back accesses.
- coherent dual-parameter controller allows atomic (to microengines) accesses to 2 parameters by the host.
- Development support features:
 - Nexus class 3 debug support (optional, associated with the eTPU-Nexus Block NDEDI).
 - Software breakpoints.
 - Debug interface supporting single-step execution, forced microinstruction execution, Hardware breakpoints and watchpoints on several conditions.
- Safety support features:
 - SCM (code memory) continuous signature-check built-in self test (MISC - Multiple Input Signature Calculator), runs concurrently with eTPU normal operation.
 - See [eTPU Plus Enhancements over eTPU](#) for additional safety features.

37.3.2.2 eTPU Enhancements over TPU3

- 32 orthogonal channels with enhanced functionality. Full support for double action with double match and double transition sub-mode combinations.

- Input and Output features separated in channel logic and microinstructions, allowing input and output signals to be processed separately or combined.
- Increased time resolution and execution unit to 24 bits.
- Increased linear code memory, shared by two eTPU Engines, configurable up to 16K positions (64 KB).
- Increased SDM address range (8 KB each Engine) and width (32 bits per parameter). The SDM can be dynamically allocated to support variable number of parameters for each channel. Each channel can have access to at least 256 parameters.
- The SDM is fully shared by two eTPU Engines (SDM), supporting direct inter-engine communication with the help of hardware semaphores.
- Enhanced arithmetic operations, including add/subtract with carry, absolute value, multiple shift and rotate, conditional execution with variable operand widths.
- Enhanced logic operations, including bitwise operations (and, or, xor) and bit manipulation, with conditional execution. Support for read-modify-write of any bit in the SDM.
- Hardware for Multiply/MAC/Divide, running in parallel to execution of other operations. The 24-bit divide result is available after 13 other unrelated instructions. Multiplication supports any data width of both operands (8, 16 or 24 bits), signed or unsigned. A 24x24 Multiply/MAC result is available after four other unrelated instructions. A 24x8 Multiply/MAC result is available after one other unrelated instruction.
- Supports export/import of time bases from other sources through the real time bus (STAC - Shared Time and Counter bus). This internal bus is used for sharing real time data between multiple peripherals.
- Contains angle clock hardware, supported by microcode, which can provide a 24-bit angle bus instead of time bus. This feature enables the eTPU to run angle based engine control applications.
- More interrupt types. Each eTPU channel can generate a data transfer request interrupt, in addition to regular interrupts, and one global exception interrupt. Data Transfer requests can be used either as interrupt sources or DMA requests. This feature takes advantage of DMA peripherals which off-load the Host. Interrupt Overflow status is also provided.
- Improved visibility to the Host (pin states, time bases, serviced channel).

- An edge case of priority inversion on TPU3 Scheduler was resolved.
- Supports channel link requests between eTPU Engines.

37.3.2.3 eTPU Plus Enhancements over eTPU

eTPU Plus features include the enhancements below in addition to all eTPU features

- TCR1, channel logic and digital filters (both channel and TCRCLK) now have an option to run at divisions of full eTPU clock speed, besides eTPU clock / 2 (see [T2/T4 channel timing](#)).
- Channels support unordered transitions: transition B can now be detected before transition A. Related to this enhancement, TDLA and TDLB can now be independently negated by microcode (see [Channel Mode Logic and Event Flags](#))
- Channel logic can now work on full eTPU clock, allowing faster response with slow clocks (see [T2/T4 channel timing](#)).
- Added a new User Programmable Channel Mode: the blocking, enabling, service request and capture characteristics of this channel mode can be programmed via microcode (see [General Channel registers](#)).
- Microinstructions now provide an option to issue Interrupt and Data Transfer requests selected by CHAN. They can also be requested simultaneously at the same instruction (see [Channel Interrupt and Data Transfer Requests](#)).
- Channel Flags 0 and 1 can now be tested for branching, besides selecting the entry point (see [Branch Operations](#)).
- Channel digital filters can be bypassed, and can be clocked by full eTPU clock (see [Bypass Mode](#)).

Backward compatibility is maintained so that all legacy object code (both Host and microcode) runs on eTPU Plus without modification.

37.3.2.4 eTPU2 enhancements over eTPU Plus

eTPU2 features include the enhancements below in addition to all eTPU Plus enhancements and eTPU features.

- Scheduler priority-passing mechanism can now be disabled (see [Priority passing disabling](#)).
- New Watchdog mechanism kills threads over a programmable timeout (see [Watchdog](#)).
- New counter allows microengine load information collection for performance analysis (see [Idle Counter](#)).
- Channels 1 and 2 (besides channel 0) can now be selected to control the EAC (see the TBCR_ENG1 and TBCR_ENG2 eTPU Time Base Configuration Registers).
- Timebase prescalers are now reset when the GTBE input is negated, guaranteeing synchronization with eMIOS in all cases (see [GTBE - Global Time Base Enable](#)).
- New MISC flag indicates when an SCM signature calculation round is completed. This allows measuring of the average MISC scan period in a real application situation (see the eTPU Module Configuration Register - MCR).
- New channel TCCEA flag allows continuous capture even after TDLA is set, making it fully compatible with TPU behavior (see [TCCEA - Transition Continuous Capture Enable](#)).
- New branch condition PRSS tells the pin state at the time when a channel (match or transition) service request occurred (see [Pin Control Registers](#)).
- MRLEA/B can now be negated independently by microcode (see [Disable Matches](#)).
- New Engine Relative address mode allows a function to access SDM address space common to one engine, but distinct between engines (see Engine Relative Addressing Mode under [SDM Addressing Modes](#)).
- Memory error support for Code (SCM) and Data (SDM) memories: error detection, correction, and hardware-assisted soft-error fixing (see [Memory error support](#)).
- Sharing of microengine and host accesses to subsystem modules (for instance, the Reaction Module), allowing their control by the eTPU, the host, or both (see [Shared Subsystems support](#)).

Backward compatibility is maintained so legacy object code (both Host and microcode) runs on eTPU2 without modification.

37.3.2.5 eTPU2 Plus enhancements over eTPU2

eTPU2 Plus features include the enhancements listed below in addition to all eTPU features, eTPU Plus enhancements and eTPU2 enhancements.

- New Latency Monitor allows latency measurements, either for individual channels or channels combined in a single priority (High, Medium or Low).
- Watchdog can now be configured to disable thread forced ending and channel blocking, allowing it to monitor thread lengths with no interference on the application.

- EAC Angle Tick Generator can now be clocked by the TCR2 prescaler output, as an alternative to the TCR1 clock.
- EAC now supports up to 7 missing teeth on hardware.

Backward compatibility is maintained so legacy object code (both Host and microcode) runs on eTPU2 Plus without modification.

37.3.3 Modes of operation

eTPU can be seen as capable of working in the following modes:

- **User Configuration Mode**

User has the ability to program the eTPU Cores with User Time Functions, having access to the Shared Code Memory (SCM).

- **User Mode**

User does not access the eTPU Shared Code Memory:

- Use of predefined eTPU Functions
- No need for eTPU Core programming ability

- **Debug Mode**

User debugs eTPU code, accessing special Trace/Debug features via Nexus interface:

- hardware breakpoint/watchpoint setting
- access to internal registers
- single-step execution
- forced instruction execution
- software breakpoint insertion and removal.

- **Module Disable Mode**

eTPU Engine clocks are stopped through a register write to the MDIS field in [eTPU Engine 1 Configuration Register \(eTPU_SCR_ECR_ENG1\)](#) and [eTPU Engine 2 Configuration Register \(eTPU_SCR_ECR_ENG2\)](#), saving power. Input sampling stops. eTPU Engines can be in Module Disable Mode independently. Module Disable Mode stops only the Engine clock, so that the Shared BIU, and Global

Channel registers can be accessed, and interrupts and DMAs can be cleared and enabled/disabled. An Engine only enters Module Disable Mode when any currently running thread is finished (see [Functions and threads](#)).

- **Stop Mode**

Stop Mode is entered when eTPU answers device stop request assertion with stop acknowledge. The definition of which clocks are stopped is made at the MCU level, which defines whether or not registers can be accessed, interrupts and DMA requests cleared.

These modes are loosely selected: there is no unique register field or signals to choose between them. Some features of one mode can be used with features of other mode(s). More on this subject can be found on [eTPU mode selection](#) below.

Note

Throughout this chapter, an engine is said to be "stopped" if it is either in Module Disable mode or Stop mode.

37.3.3.1 eTPU mode selection

User and User Configuration are the production operating modes, and differ from each other only in access to SCM. User programmability is only possible with a RAM SCM.

On chips where the SCM is implemented as a RAM, it can either be accessed directly from IP-Bus for code loading, or for software breakpoint setting. On chips with a ROM SCM, an internal SCM Emulation RAM may be used, depending on the specific MCU implementation, to replace ROM SCM for test or debug purposes. SCM Emulation RAM is selected in a MCU-specific way. For more details, see [SCM Emulation](#).

Debug Mode is characterized by the use of the debug interface features. Specifically, Nexus blocks provide Nexus class 3 debug features. For more information on debug features, refer to [Development support features](#).

Module Disable Mode is entered by setting the MDIS field in [eTPU Engine 1 Configuration Register \(eTPU_SCR_ECR_ENG1\)](#) and [eTPU Engine 2 Configuration Register \(eTPU_SCR_ECR_ENG2\)](#). eTPU Engines can be individually stopped going into Module Disable Mode (there is one eTPU_SCR_ECR register for each Engine). Each engine can leave Module Disable Mode by writing MDIS=0 (which can only be done if VIS=0).

Stop Mode is activated by IP-Bus (device stop request). In this case, the eTPU waits for both eTPU Engines to enter in stop mode, and then asserts the stop acknowledge line. eTPU leaves Stop Mode when device stop request is negated, but only if VIS=0. If device stop request is negated and VIS=1, eTPU will leave Stop Mode as soon as VIS=0.

Note

An Engine can stay in Module Disable mode when it leaves Stop Mode if its bit MDIS=1, even if the other leaves it.

37.4 Detailed signal descriptions

Detailed signal descriptions are presented in the following sections.

37.4.1 eTPU channel output signals

Each channel output signal is associated with a channel. The microcode may affect the logic level of an output signal¹ by implementing one of two actions:

- Specify the logic level output to the signal when there is a match or a transition.
- Immediately force a logic level.

The output signal may also be forced to a logic level, independently of the output value from the channel logic, by one of the four (each Engine) output disable input signals (see [eTPU channel output disable signals](#)).

The output buffer enable signals are not used.

37.4.2 eTPU channel input signals

Each channel input signal is associated with a channel. The microcode can directly control the effect of the transition edge. Each channel can be programmed to sense a transition when a rising and/or falling edge is detected. The channel logic can also process two transition events, and relate these events to each other and to other programmed timer events. The edge sensitivities of the two transition events are configured independently by microcode. For further information refer to [Enhanced Channels](#) and [Transition Detection and Pin Action Control](#).

1. Note that the minimum pulse width is one microcycle (two eTPU clocks), and slow pads may not be able to transfer it on time. For generation of very short pulses the eTPU pads have to be programmed by the system integration for fast operation mode with the voltage levels defined for fast pad operation in the MCU technology.

Each channel input signal has an associated synchronizer made of two flip-flops sampling the signal on every other eTPU clock², followed by a digital filter. This digital filter can work in three sub-modes, whose purpose is to filter out noise pulses that have width less than a programmed value of eTPU clocks, preventing these transitions from being input to the transition detect logic. The synchronizer and digital filter are guaranteed to pass pulses that are greater than a programmed value. All channel input filters in one Engine work on the same mode and sampling clock. For more information on channel input filters, refer to [Enhanced Digital Filter - EDF](#). In one of the Angle Modes, the output of the digital filter of channel 0 is replaced by the output of TCRCLK signal digital filter (see [EAC - eTPU Angle Counter](#)).

37.4.3 Time Base Clock Signal — TCRCLK

TCRCLK is an input signal used to control the Time Bases TCR1 and TCR2. There is one independent TCRCLK input for each Engine. For pulse accumulator operations TCRCLK can be used as a gate for a counter based on the eTPU clock divided by eight. For Angle operations TCRCLK can be used to get the tooth transition indications in Angle Mode. Further details can be found on [Time bases](#) and [EAC - eTPU Angle Counter](#).

Like the channel input signals, the TCRCLK signal has an associated synchronizer followed by a digital filter. This digital filter can work in two sub-modes, whose purpose is to filter out noise pulses that have width less than a programmed value of eTPU clocks, preventing these transitions from being input to the transition detect logic. The synchronizer and digital filter are guaranteed to pass pulses that are greater than a programmed value. The clock and operation sub-mode of the TCRCLK filter is configured independently of the other channel input filters, through the TCRCF field in [eTPU Engine 1 Time Base Configuration Register \(eTPU_TBR_TBCR_ENG1\)](#) and [eTPU Engine 2 Time Base Configuration Register \(eTPU_TBR_TBCR_ENG2\)](#). For more information on filter sub-modes, see [TCRCLK digital filter](#). In one of the Angle Modes, the output of the digital filter of channel 0 is replaced with the output of TCRCLK signal digital filter (see [EAC - eTPU Angle Counter](#)).

37.4.4 eTPU channel output disable signals

Each of these 4 input signals are used to force the outputs of a group of 8 channels to an inactive level. When an ODIS input is active, all the channels in its group of 8 that have their fields ODIS=1 in [eTPU Channel n Configuration Register for Engine 1 \(eTPU_CCCR_CnCR_ENG1\)](#) and [eTPU Channel n Configuration Register for Engine 2](#)

2. Sampled on the T4 microcycle phase, see [Microcycle and I/O timing](#).

([eTPU_CCCR_CnCR_ENG2](#)) have their outputs forced to the opposite of the value specified in bit OPOL of the same register. Therefore, channels can be individually selected to be affected by the output disable signals, as well as their disabling forced polarity (see 'Pin State Input/Output Logic' figure in [Pin Control Registers](#)).

See the chip-specific eTPU information for details on the output channel disable group definition.

In a Dual-engine eTPU there are 8 output disable signals for the 64 channels.

37.5 Memory map/register definition

The guideline for the description of all bits and fields throughout the “Memory Map/ Register Definition” section is to provide only a brief explanation (without examples or method of use) of the features, since it will be used mainly as a reference for the reader that is studying the “Functional Description” section, where those features are explained in detail.

37.6 Memory map

The eTPU System simplified memory map is shown in [Table 37-4](#). Each of the register areas shown may have their own reserved address areas.

[Table 37-5](#) shows a detailed memory map. Offsets are relative to the eTPU Base address, which is MCU-dependent.

See the chip-specific eTPU information about accessing eTPU registers in chip-level Stop mode.

SCM unused area is decoded and returns a fixed opcode, determined by [eTPU SCM Off-range Data Register \(eTPU_SCR_SCMOFFDATAR\)](#).

Table 37-4. High-level memory map

Offset	Use
0x00 – 0x1F	System Configuration Registers
0x20 – 0x2F	eTPU 1 Time Base Registers
0x30 – 0x3F	Reserved ¹
0x40 – 0x4F	eTPU 2 Time Base Registers
0x50 – 0x5F	Reserved ¹
0x60 – 0x6F	eTPU 1 Extra Engine Registers
0x70 – 0x7F	eTPU 2 Extra Engine Registers

Table continues on the next page...

Table 37-4. High-level memory map (continued)

Offset	Use
0x80 – 0xFF	Latency Monitor Registers
0x100 – 0x13F	Memory Error Support Registers
0x140 – 0x1FF	Reserved ¹
0x200 – 0x2FF	eTPU 1/2 Global Channel Registers
0x300 – 0x3FF	Reserved ¹
0x400 – 0x7FF	eTPU 1 Channel Registers
0x800 – 0xBFF	eTPU 2 Channel Registers
0xC00 – 0xFFFF	Shared Subsystems Integration Registers
0x1000 – 0x3FFF	Reserved ¹
0x4000 – 0x7FFF	SSS - Shared Subsystems area
0x8000 – 0xBFFF ²	SDM - Shared Data Memory
0xC000 – 0xFFFF ²	SDM PSE mirror ³
0x10000 – 0x1FFFF ²	SCM - Shared Code Memory ⁴

1. Reserved addresses must not be used. Access to these memory positions complete with 0-wait-states, but the result is unspecified.
2. Actual sizes of SCM and SDM are MCU-dependent (see the chip-specific eTPU information).
3. Parameter Sign Extension access area, see [Parameter Access](#).
4. SCM access is available only when bit VIS=1 on register eTPU_SCR_MCR, under certain conditions (see [eTPU Module Configuration Register \(eTPU_SCR_MCR\)](#)).

Table 37-5. Detailed memory map

Offset	Use
System Configuration Registers	
0x00	eTPU_SCR_MCR - eTPU Module Configuration Register
0x04	eTPU_SCR_CDCR - eTPU Coherent Dual-Parameter Controller Register
0x08	Reserved
0x0C	eTPU_SCR_MISCCMPR - eTPU MISC Compare Register
0x10	eTPU_SCR_SCMOFFDATAR - eTPU SCM Off-range Data Register ¹
0x14	eTPU_SCR_ECR_ENG1 - eTPU 1 Engine Configuration Register
0x18	eTPU_SCR_ECR_ENG2 - eTPU 2 Engine Configuration Register
0x1C	Reserved
Time Base Registers	
0x20	eTPU_TBR_TBCR_ENG1 - eTPU 1 Time Base Configuration Register
0x24	eTPU_TBR_TB1R_ENG1 - eTPU 1 Time Base 1
0x28	eTPU_TBR_TB2R_ENG1 - eTPU 1 Time Base 2
0x2C	eTPU_TBR_STAC_ENG1 - eTPU 1 STAC Configuration Register
0x30	Reserved
0x34	Reserved
0x38	Reserved
0x3C	Reserved

Table continues on the next page...

Table 37-5. Detailed memory map (continued)

Offset	Use
0x40	eTPU_TBR_TBCR_ENG2 - eTPU 2 Time Base Configuration Register
0x44	eTPU_TBR_TB1R_ENG2 - eTPU 2 Time Base 1
0x48	eTPU_TBR_TB2R_ENG2 - eTPU 2 Time Base 2
0x4C	eTPU_TBR_STAC_ENG2 - eTPU 2 STAC Configuration Register
0x50	Reserved
0x54	Reserved
0x58	Reserved
0x5C	Reserved
Engine Related Registers	
0x60	eTPU_ER_WDTR_ENG1 - eTPU 1 Watchdog Timer Register
0x64	Reserved
0x68	eTPU_ER_IDLER_ENG1 - eTPU 1 Idle Counter Register
0x6C	Reserved
0x70	eTPU_ER_WDTR_ENG2 - eTPU 2 Watchdog Timer Register
0x74	Reserved
0x78	eTPU_ER_IDLER_ENG2 - eTPU 2 Idle Counter Register
0x7C	Reserved
Latency Monitor Registers	
0x80	eTPU_LMR_LM0CR_ENG1 - eTPU 1 Latency Monitor 0 Configuration Register
0x84	eTPU_LMR_LM0MR_ENG1 - eTPU 1 Latency Monitor 0 Measure Register
0x88	eTPU_LMR_LM1CR_ENG1 - eTPU 1 Latency Monitor 1 Configuration Register
0x8C	eTPU_LMR_LM1MR_ENG1 - eTPU 1 Latency Monitor 1 Measure Register
0xC0	eTPU_LMR_LM0CR_ENG2 - eTPU 2 Latency Monitor 0 Configuration Register
0xC4	eTPU_LMR_LM0MR_ENG2 - eTPU 2 Latency Monitor 0 Measure Register
0xC8	eTPU_LMR_LM1CR_ENG2 - eTPU 2 Latency Monitor 1 Configuration Register
0xCC	eTPU_LMR_LM1MR_ENG2 - eTPU 2 Latency Monitor 1 Measure Register
Memory Error Support Registers ²	
0x100	eTPU_MESR_MECR - eTPU Memory Error Control Register
0x104	eTPU_MESR_DEIAR - eTPU Data Error Injection Address Register
0x108	eTPU_MESR_DEIDPR - eTPU Data Error Injection Data Pattern Register
0x10C	eTPU_MESR_DEIPPR - eTPU Data Error Injection Parity Pattern Register
0x110	eTPU_MESR_DERAR - eTPU Data Error Report Address Register
0x114	eTPU_MESR_DERDR - eTPU Data Error Report Data Register
0x118	eTPU_MESR_DERSR - eTPU Data Error Report Syndrome Register
0x11C	Reserved
0x120	eTPU_MESR_MESR - eTPU Memory Error Status Register
0x124	eTPU_MESR_CEIAR - eTPU Code Error Injection Address Register
0x128	eTPU_MESR_CeidPR - eTPU Code Error Injection Data Pattern Register
0x12C	eTPU_MESR_CeiPPR - eTPU Code Error Injection Parity Pattern Register

Table continues on the next page...

Table 37-5. Detailed memory map (continued)

Offset	Use
0x130	eTPU_MESR_CERAR - eTPU Code Error Report Address Register
0x134	eTPU_MESR_CERDR - eTPU Code Error Report Data Register
0x138	eTPU_MESR_CERSR - eTPU Code Error Report Syndrome Register
0x13C	eTPU_MESR_CEFR - eTPU Code Error Fix Register
0x140 - 0x1FF	Reserved
Global Channel Registers	
0x200	eTPU_GCR_CISR_ENG1 - eTPU 1 Channel Interrupt Status Register
0x204	eTPU_GCR_CISR_ENG2 - eTPU 2 Channel Interrupt Status Register
0x208	Reserved
0x20C	Reserved
0x210	eTPU_GCR_CDTRSR_ENG1 - eTPU 1 Channel Data Transfer Request Status Register
0x214	eTPU_GCR_CDTRSR_ENG2 - eTPU 2 Channel Data Transfer Request Status Register
0x218	Reserved
0x21C	Reserved
0x220	eTPU_GCR_CIOSR_ENG1 - eTPU 1 Channel Interrupt Overflow Status Register
0x224	eTPU_GCR_CIOSR_ENG2 - eTPU 2 Channel Interrupt Overflow Status Register
0x228	Reserved
0x22C	Reserved
0x230	eTPU_GCR_CDTROSR_ENG1 - eTPU 1 Channel Data Transfer Request Overflow Status Register
0x234	eTPU_GCR_CDTROSR_ENG2 - eTPU 2 Channel Data Transfer Request Overflow Status Register
0x238	Reserved
0x23C	Reserved
0x240	eTPU_GCR_CIER_ENG1 - eTPU 1 Channel Interrupt Enable Register
0x244	eTPU_GCR_CIER_ENG2 - eTPU 2 Channel Interrupt Enable Register
0x248	Reserved
0x24C	Reserved
0x250	eTPU_GCR_CDTREER_ENG1 - eTPU 1 Channel Data Transfer Request Enable Register
0x254	eTPU_GCR_CDTREER_ENG2 - eTPU 2 Channel Data Transfer Request Enable Register
0x258	Reserved
0x25C	Reserved
0x260	eTPU_GCR_WDSR_ENG1 - eTPU 1 Watchdog Status Register
0x264	eTPU_GCR_WDSR_ENG2 - eTPU 2 Watchdog Status Register
0x268-0x27F	Reserved
0x280	eTPU_GCR_CPSSR_ENG1 - eTPU 1 Channel Pending Service Status Register
0x284	eTPU_GCR_CPSSR_ENG2 - eTPU 2 Channel Pending Service Status Register
0x288	Reserved
0x28C	Reserved
0x290	eTPU_GCR_CSSR_ENG1 - eTPU 1 Channel Service Status Register
0x294	eTPU_GCR_CSSR_ENG2 - eTPU 2 Channel Service Status Register

Table continues on the next page...

Table 37-5. Detailed memory map (continued)

Offset	Use
0x298	Reserved
0x29C	Reserved
0x300 - 0x3FF	Reserved
Channel Configuration and Control Registers	
0x400	eTPU_CCCR_C0CR_ENG1 - eTPU 1 Channel 0 Configuration Register
0x404	eTPU_CCCR_C0SCR_ENG1 - eTPU 1 Channel 0 Status and Control Register
0x408	eTPU_CCCR_C0HSRR_ENG1 - eTPU 1 Channel 0 Host Service Request Register
0x40C	Reserved
0x410	eTPU_CCCR_C1CR_ENG1 - eTPU 1 Channel 1 Configuration Register
0x414	eTPU_CCCR_C1SCR_ENG1 - eTPU 1 Channel 1 Status and Control Register
0x418	eTPU_CCCR_C1HSRR_ENG1 - eTPU 1 Channel 1 Host Service Request Register
0x41C	Reserved
-----	-----
0x5F0	eTPU_CCCR_C31CR_ENG1 - eTPU 1 Channel 31 Configuration Register
0x5F4	eTPU_CCCR_C31SCR_ENG1 - eTPU 1 Channel 31 Status and Control Register
0x5F8	eTPU_CCCR_C31HSRR_ENG1 - eTPU 1 Channel 31 Host Service Request Register
0x5FC - 0x7FF	Reserved
0x800	eTPU_CCCR_C0CR_ENG2 - eTPU 2 Channel 0 Configuration Register
0x804	eTPU_CCCR_C0SCR_ENG2 - eTPU 2 Channel 0 Status and Control Register
0x808	eTPU_CCCR_C0HSRR_ENG2 - eTPU 2 Channel 0 Host Service Request Register
0x80C	Reserved
0x810	eTPU_CCCR_C1CR_ENG2 - eTPU 2 Channel 1 Configuration Register
0x814	eTPU_CCCR_C1SCR_ENG2 - eTPU 2 Channel 1 Status and Control Register
0x818	eTPU_CCCR_C1HSRR_ENG2 - eTPU 2 Channel 1 Host Service Request Register
0x81C	Reserved
-----	-----
0x9F0	eTPU_CCCR_C31CR_ENG2 - eTPU 2 Channel 31 Configuration Register
0x9F4	eTPU_CCCR_C31SCR_ENG2 - eTPU 2 Channel 31 Status and Control Register
0x9F8	eTPU_CCCR_C31HSRR_ENG2 - eTPU 2 Channel 31 Host Service Request Register
0x9FC - 0xBFF	Reserved
Shared Subsystem Support Registers	
0xC00	eTPU_SSIR_SSS0IR - eTPU SSS 0 Integration Register
0xC04	eTPU_SSIR_SSS1IR - eTPU SSS 1 Integration Register
0xC08	eTPU_SSIR_SSS2IR - eTPU SSS 2 Integration Register
0xC0A	eTPU_SSIR_SSS3IR - eTPU SSS 3 Integration Register
0xC0C	eTPU_SSIR_SSS4IR - eTPU SSS 4 Integration Register
0xC10	eTPU_SSIR_SSS5IR - eTPU SSS 5 Integration Register
0xC14	eTPU_SSIR_SSS6IR - eTPU SSS 6 Integration Register
0xC18	eTPU_SSIR_SSS7IR - eTPU SSS 7 Integration Register

Table continues on the next page...

Table 37-5. Detailed memory map (continued)

Offset	Use
0xC1A	eTPU_SSIR_SSS8IR - eTPU SSS 8 Integration Register
0xC1C	eTPU_SSIR_SSS9IR - eTPU SSS 9 Integration Register
0x1000-0x3FFF	Reserved
0x4000 - 0x7FFF	Shared Subsystems area - SSS
0x8000 - 0xBFFF ³	Shared Data Memory - SDM
0xC000 - 0xFFFF ³	Shared Data Memory - SDM - PSE mirror ⁴
0x10000-1FFFF ⁵	Shared Code Memory - SCM ⁶

1. This register is not implemented in some MCUs; see [eTPU SCM Off-range Data Register \(eTPU_SCR_SCMOFFDATAR\)](#).
2. These registers are not implemented in some MCUs; see [Memory error support](#).
3. The actual SDM size is MCU-dependent (see the chip-specific eTPU information). SDM is allocated from the starting value of the given range of offsets.
4. Parameter Sign Extension access area, see [Parameter Access](#).
5. The actual SCM size is MCU-dependent (see the chip-specific eTPU information). SCM is allocated from the starting value of the given range of offsets. When the size is not the maximum, the unused SCM address range returns the value of [eTPU SCM Off-range Data Register \(eTPU_SCR_SCMOFFDATAR\)](#).
6. SCM access is available only when eTPU_SCR_MCR[VIS]=1, under certain conditions (see [eTPU Module Configuration Register \(eTPU_SCR_MCR\)](#)).

37.7 System Configuration memory map and registers

eTPU_SCR memory map

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
0	eTPU Module Configuration Register (eTPU_SCR_MCR)	32	R/W	See section	37.7.1/1502
4	eTPU Coherent Dual-Parameter Controller Register (eTPU_SCR_CDCR)	32	R/W	0000_0000h	37.7.2/1506
C	eTPU MISC Compare Register (eTPU_SCR_MISCCMPR)	32	R/W	0000_0000h	37.7.3/1508
10	eTPU SCM Off-range Data Register (eTPU_SCR_SCMOFFDATAR)	32	R/W	F377_5FFBh	37.7.4/1508
14	eTPU Engine 1 Configuration Register (eTPU_SCR_ECR_ENG1)	32	R/W	See section	37.7.5/1509
18	eTPU Engine 2 Configuration Register (eTPU_SCR_ECR_ENG2)	32	R/W	See section	37.7.6/1513

37.7.1 eTPU Module Configuration Register (eTPU_SCR_MCR)

This register is global to both eTPU engines, and resides in the shared BIU. It gathers global configuration and status in the eTPU system, including global exception. It is also used for configuring the SCM (shared code memory) operation and test.

Address: 0h base + 0h offset = 0h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	SDMERR	WDTO1	WDTO2	MGE1	MGE2	ILF1	ILF2	SSAERR	0	0	SCMSIZE				
W	GEC															
Reset	0	0	0	0	0	0	0	0	0	0	0	*	*	*	*	*

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	LTO1	LTO2	0		SCMMISC	SCMMISF	SCMMISEN	SCMERR	0	VIS	0				GTBE	
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

* Notes:

- SCMSIZE field: See the chip-specific eTPU information for this field's reset value.

eTPU_SCR_MCR field descriptions

Field	Description
0 GEC	Global Exception Clear. This write-only bit negates global exception request and clears global exception status bits SDMERR, SSAERR, SCMERR, WDTO1/2, MGE1/2, ILF1/2, LTO1/2, and SCMMISF. GEC works the same way with either one or both Engines in Module Disable Mode. 0 Keep Global Exception request and status bits SDMERR, SSAERR, SCMERR, WDTO1/2, ILF1/2, MGE1/2, LTO1/2, and SCMMISF as is. 1 Negate Global Exception, clear status bits SDMERR, SSAERR, SCMERR, WDTO1/2, ILF1/2, MGE1/2, LTO1/2, and SCMMISF.
1 SDMERR	SDM Read Error This flag indicates that an SDM non-correctable read error occurred on a microengine or host (including CDC) read, generating a Global Exception. This bit is cleared by writing 1 to GEC. 0 No Global Exception pending because of SDM read error. 1 Global Exception requested by SDM read error is pending.
2 WDTO1	Watchdog 1 Timeout. Flag WDTO1 indicates that a Watchdog Timeout occurred in engine 1, generating a Global Exception. This bit is cleared by writing 1 to GEC. 0 No Global Exception pending because of Watchdog timeout. 1 Global Exception requested by Watchdog timeout.
3 WDTO2	Watchdog 2 Timeout. Flag WDTO2 indicates that a Watchdog Timeout occurred in engine 2, generating a Global Exception. This bit is cleared by writing 1 to GEC. 0 No Global Exception pending because of Watchdog timeout. 1 Global Exception requested by Watchdog timeout.
4 MGE1	Microcode Global Exception Engine 1. This bit indicates that a global exception was asserted by microcode executed on engine 1. The determination of the reason why the global exception was asserted is application dependent: it can be coded in an SDM status parameter, for instance. This bit is cleared by writing 1 to GEC. 0 No microcode-requested global exception pending. 1 Global exception requested by microcode is pending.
5 MGE2	Microcode Global Exception Engine 2. This bit indicates that a global exception was asserted by microcode executed on engine 2. The determination of the reason why the global exception was asserted is application dependent: it can be coded in an SDM status parameter, for instance. This bit is cleared by writing 1 to GEC. 0 No microcode-requested global exception pending. 1 Global exception requested by microcode is pending.
6 ILF1	Illegal Instruction Flag for Engine 1. The ILFA1 flag is set by the microengine to indicate that an illegal instruction was decoded in Engine 1. This bit is cleared by host writing 1 to GEC. See Illegal instructions for more details.

Table continues on the next page...

eTPU_SCR_MCR field descriptions (continued)

Field	Description
	0 Illegal Instruction not detected. 1 Illegal Instruction detected by Engine 1.
7 ILF2	Illegal Instruction Flag for Engine 2. The ILFA2 flag is set by the microengine to indicate that an illegal instruction was decoded in Engine 2. This bit is cleared by host writing 1 to GEC. See Illegal instructions for more details. 0 Illegal Instruction not detected. 1 Illegal Instruction detected by Engine 2.
8 SSAERR	Shared Subsystem Access Error. This flag indicates that a bus error occurred in an microengine access to the Shared Subsystem Area (see Shared Subsystems support). This bit is cleared by writing 1 to GEC. NOTE: Bus errors on host accesses to the Shared Subsystem Area do not set this bit neither asserts the Global Exception, but the bus error is propagated to the host CPU 0 No bus error returned on a microengine Shared Subsystem Area access. 1 Bus error returned on a microengine Shared Subsystem Area access.
9 Reserved	Reserved. This field is reserved. This read-only field is reserved and always has the value 0.
10 Reserved	Reserved. This field is reserved. This read-only field is reserved and always has the value 0.
11–15 SCMSIZE	SCM Size This read-only field holds the number of 2 KB SCM Blocks minus 1. For more information, see the MCU Reference Manual. See the chip-specific eTPU information for this field's reset value.
16 LTO1	Latency Timeout for Engine 1. The LTO1 flag indicates the latency measured by the Latency Monitor of Engine 1 surpassed the specified maximum. See Latency Monitor for details. This bit is cleared by writing 1 to GEC. 0 No latency timeout detected by Engine 1's Latency Monitor. 1 Latency timeout detected by Engine 1's Latency Monitors.
17 LTO2	Latency Timeout for Engine 2. The LTO2 flag indicates the latency measured by the Latency Monitor of Engine 2 surpassed the specified maximum. See Latency Monitor for details. This bit is cleared by writing 1 to GEC. 0 No latency timeout detected by Engine 2's Latency Monitor. 1 Latency timeout detected by Engine 2's Latency Monitors.
18–19 Reserved	Reserved. This field is reserved. This read-only field is reserved and always has the value 0.

Table continues on the next page...

eTPU_SCR_MCR field descriptions (continued)

Field	Description
20 SCMMISC	SCM MISC Complete flag. Flag SCMMISC indicates that MISC has completed the evaluation of the SCM signature since reset or since the last time it was cleared. SCMMISC is cleared by writing 1 to it, and is not cleared when MISC is disabled (SCMMISEN=0). SCMMISC asserts at the end of the SCM memory scan, either if the signature matches or not. 0 MISC has not yet completed an SCM signature calculation and compare since the last time SCMMISC was cleared. 1 MISC completed at least one SCM signature calculation and compare since the last time SCMMISC was cleared.
21 SCMMISF	SCM MISC Flag. The SCMMISF bit is set by the SCM MISC (Multiple Input Signature Calculator) logic to indicate that the calculated signature does not match the expected value, at the end of a MISC iteration. See SCM Test - Multiple Input Signature Calculator for more details. This bit is cleared when Global Exception is cleared by writing 1 to GEC. 0 Signature mismatch not detected. 1 MISC has read entire SCM array and the expected signature in eTPU MISC Compare Register (eTPU_SCR_MISCCMPR) does not match the value calculated.
22 SCMMISEN	SCM MISC Enable. The SCMMISEN bit is used for enabling/disabling the operation of the MISC logic. SCMMISEN is readable and writable at any time. The MISC logic will only operate when this bit is set to 1. When the bit is reset, the MISC address counter is set to the initial SCM address. When enabled, the MISC will continuously cycle through the SCM addresses, reading each and calculating a CRC. In order to save power, the MISC can be disabled by clearing the SCMMISEN bit. See SCM Test - Multiple Input Signature Calculator for more details. SCMMISEN resets automatically when MISC logic detects an error, i.e., when SCMMISF transitions from 0 to 1, disabling the MISC operation. 0 MISC operation disabled. The MISC logic is reset to its initial state. 1 MISC operation enabled.
23 SCMERR	SCM Read Error. This flag indicates that an SCM non-correctable read error occurred on a host or microengine (including MISC) read, generating a Global Exception. This bit is cleared by writing 1 to GEC. 0 No Global Exception pending because of SCM read error. 1 Global Exception requested by SCM read error is pending.
24 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
25 VIS	SCM Visibility Bit. VIS bit turns SCM visible to the IP-Bus and resets MISC state (but SCMMISEN keeps its value). This bit is write protected when any of the Engines is not halted or stopped. When VIS=1, the MDIS bits in the eTPU Memory Error Support registers are write protected.

Table continues on the next page...

eTPU_SCR_MCR field descriptions (continued)

Field	Description
	0 SCM is not visible to the slave bus. Accessing SCM address space issues a bus error, writes are protected and reads are meaningless. 1 SCM is visible to the slave bus. MISC state is reset.
26–30 Reserved	Reserved. This field is reserved. This read-only field is reserved and always has the value 0.
31 GTBE	Global Time Base Enable. GTBE enables time bases in both Engines, allowing them to be started synchronously. NOTE: Global Time Base Enable action may also depend on other blocks, as explained in GTBE - Global Time Base Enable . NOTE: When GTBE is turned off with Angle Mode enabled, the EAC must be reinitialized before GTBE is turned on again. The EAC reinitialization procedure is described in Restarting angle logic . 0 Time bases in both Engines are disabled to run. 1 Time bases in both Engines are enabled to run.

37.7.2 eTPU Coherent Dual-Parameter Controller Register (eTPU_SCR_CDCR)

This register configures and controls dual-parameter coherent transfers. For more information, see [Coherent Dual-parameter Controller \(CDC\)](#).

Address: 0h base + 4h offset = 4h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	STS		CTBASE				PBBASE									
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	PWIDTH		PARM0						WR	PARM1						
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

eTPU_SCR_CDCR field descriptions

Field	Description
0 STS	Start Bit. This bit is set by the host in order to start the data transfer between the parameter buffer pointed by PBBASE and the target addresses selected by the concatenation of fields CTBASE and PARM0/1. The host receives wait-states until the data transfer is complete, when this bit is reset by the coherency logic (see Coherent Dual-parameter Controller (CDC)). Therefore, host always reads STS as 0. 0 Write does not start a coherent transfer. 1 Write starts a coherent transfer.
1–5 CTBASE	Channel Transfer Base. This field concatenates with fields PARM0 and PARM1 to determine the absolute word offset (from the SDM base) of the parameters to be transferred: Parameter 0 word address = {CTBASE, PARM0} + SDM base word address. Parameter 1 word address = {CTBASE, PARM1} + SDM base word address.
6–15 PBBASE	Parameter Buffer Base Address. This field points to the base address of the parameter buffer location, with granularity of 2 parameters (8 bytes). The host (byte) address of the first parameter in the buffer is PBBASE * 8 + SDM Base Byte Address. The microengine absolute (word) address of the first parameter in the buffer is PBBASE * 2.
16 PWIDTH	Parameter Width Selection. This bit selects the width of the parameters to be transferred between the PB and the target address. 0 Transfer 24-bit parameters. The upper byte remains unchanged in the destination address. 1 Transfer 32-bit parameters. All 32 bits of the parameters are written in the destination address.
17–23 PARM0	Channel Parameter number 0. The PARM0 and PARM1 fields in concatenation with CTBASE determine the word address offset (from the SDM base) of the parameters that are destination or source (defined by WR) of the coherent transfer. The word SDM address offset of the parameters are {CTBASE, PARM0/1}. NOTE: PARM0 and PARM1 allow non-contiguous parameters to be transferred coherently. The parameter pointed by {CTBASE, PARM0} is the first transferred.
24 WR	Read/Write selection. This bit selects the direction of the coherent data transfer. 0 Read operation. Data transfer is from the selected SDM address to the PB. 1 Write operation. Data transfer is from the PB to the selected SDM address.
25–31 PARM1	Channel Parameter number 1. The PARM0 and PARM1 fields in concatenation with CTBASE determine the word address offset (from the SDM base) of the parameters that are destination or source (defined by WR) of the coherent transfer. The word SDM address offset of the parameters are {CTBASE, PARM0/1}. NOTE: PARM0 and PARM1 allow non-contiguous parameters to be transferred coherently. The parameter pointed by {CTBASE, PARM0} is the first transferred.

37.7.3 eTPU MISC Compare Register (eTPU_SCR_MISCCMPR)

This register holds the 32-bit signature expected from the whole SCM array. This register must be written by the host with the 32-bit word to be compared against the calculated signature at the end of the MISC cycle. This register is global to both eTPU Engines. For more detail, see [SCM Test - Multiple Input Signature Calculator](#).

Address: 0h base + Ch offset = Ch

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	ETPUMISCCMP																															
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	

eTPU_SCR_MISCCMPR field descriptions

Field	Description
0–31 ETPUMISCCMP	Expected Multiple Input Signature Register value. For details, see SCM Test - Multiple Input Signature Calculator .

37.7.4 eTPU SCM Off-range Data Register (eTPU_SCR_SCMOFFDATAR)

This register holds the 32-bit value returned when SCM array is accessed at non-implemented addresses, either by the Host or by the microengine. This register can be written by the host with the 32-bit instruction to be executed by the microengine to recover from runaway code. This register is global to both eTPU Engines. For more detail see [SCM Off-range data](#).

Address: 0h base + 10h offset = 10h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	ETPUSCMOFFDATAR																															
W																																
Reset	1	1	1	1	0	0	1	1	0	1	1	1	0	1	1	1	0	1	0	1	1	1	1	1	1	1	1	1	1	0	1	1

eTPU_SCR_SCMOFFDATAR field descriptions

Field	Description
0–31 ETPUSCMOFFDATAR	SCM Off-range read data value. See SCM Off-range data .

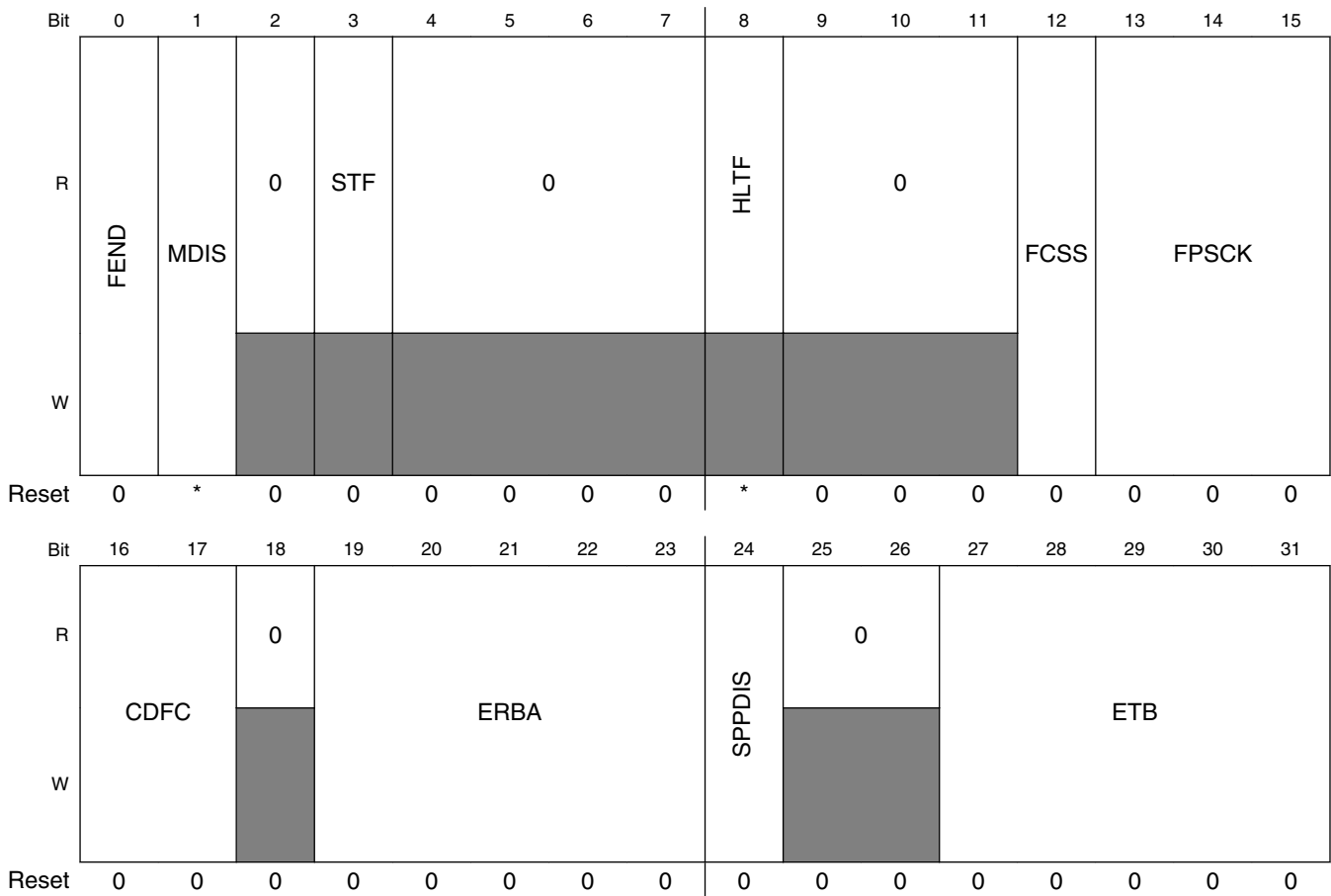
37.7.5 eTPU Engine 1 Configuration Register (eTPU_SCR_ECR_ENG1)

This register holds configuration and status fields for engine 1.

NOTE

See Register Reset Values in the chip-specific eTPU information.

Address: 0h base + 14h offset = 14h



* Notes:

- HLTF field: HLTF field reset value is 1 if, after a reset, the eTPU Nexus debug support is on and a breakpoint request is asserted. Otherwise the reset value is 0.
- MDIS field: Reset value is 0.

eTPU_SCR_ECR_ENG1 field descriptions

Field	Description
0 FEND	Force End. FEND assertion terminates any current running thread as if an END instruction have been executed (see Ending Current Thread - END).

Table continues on the next page...

eTPU_SCR_ECR_ENG1 field descriptions (continued)

Field	Description
	This bit is self-negating when the thread ends. NOTE: Only on rare occasions (e.g., during a long stall, see Microengine Stall) FEND can be read as 1, because it negates as soon as the end begins execution. Writing FEND = 1 is ignored and FEND stays 0 when the microengine is in TST, halted, stopped, or idle (no thread executing). 0 Normal operation. 1 Ends any ongoing thread.
1 MDIS	Module Disable Bit. When MDIS is set, the Engine shuts down its internal clocks, going into Module Disable Mode. TCR1 and TCR2 cease to increment, and input sampling stops. The Engine asserts the stop flag (STF) bit to indicate that it has stopped. However, the BIU continues to run, and the Host can access all registers except for the channel registers (see channel registers starting with eTPU Channel n Configuration Register for Engine 1 (eTPU_CCCR_CnCR_ENG1)) NOTE: The Timebase registers can still be read with MDIS=1, but writes are ineffective and a Bus Error is issued. Global Channel Registers and SDM can be accessed normally. After MDIS is set, even before STF asserts, data read from the channel registers is not meaningful and writes are ineffective, issuing a Bus Error. When the MDIS bit is asserted while the microcode is executing, the eTPU will stop when the thread is complete. Stop completes on the next eTPU clock after the stop condition is valid. NOTE: Once MDIS is switched from 1 to 0 or vice versa, it must not write a different value until STF changes accordingly. The MDIS bit is write-protected when VIS=1. See Register Reset Values in the chip-specific eTPU information. 0 eTPU Engine runs. 1 Commands Engine to stop its clocks.
2 Reserved	Reserved. This field is reserved. This read-only field is reserved and always has the value 0.
3 STF	Stop Flag Bit. Each Engine asserts its stop flag (STF) to indicate that it has stopped. Only then the host can assume that the Engine has actually stopped. The eTPU system is fully stopped when the STF bits of both eTPU Engines are asserted. In case of a stop request, the eTPU responds by acknowledging stop only after both eTPU 1 and eTPU 2 have been stopped. Engine only stops when any ongoing thread is complete also in this case. Summarizing Engine stop conditions, which STF reflects: STF_1 := (after stop completed) MDIS_1 device stop request. STF_2 := (after stop completed) MDIS_2 device stop request. STF_1 and STF_2 mean STF bit from engine 1 and STF bit from engine 2, respectively. See Register Reset Values in the chip-specific eTPU information.

Table continues on the next page...

eTPU_SCR_ECR_ENG1 field descriptions (continued)

Field	Description																														
	0 Engine is operating. 1 Engine has stopped (after the local MDIS bit has been asserted, or after the IP-Bus stop line has been asserted).																														
4–7 Reserved	Reserved. This field is reserved. This read-only field is reserved and always has the value 0.																														
8 HLTF	Halt Mode Flag. If eTPU Engine entered halt state, this flag is asserted. The flag remains asserted while the microengine is in halt state, even during a single-step or forced instruction execution. See Development support features for further details about entering Halt Mode. 0 eTPU Engine is not halted. 1 eTPU Engine is halted.																														
9–11 Reserved	Reserved. This field is reserved. This read-only field is reserved and always has the value 0.																														
12 FCSS	Filter Clock Source Selection Speeds up the filter clock source before the prescaler, allowing more input capture resolution at minimum prescaling. NOTE: FCSS=1 also makes the channel work on T2/T4 timing mode (see T2/T4 channel timing). 0 Use eTPU clock / 2 as EDF clock source before prescaler. 1 Use eTPU clock as EDF clock source before prescaler																														
13–15 FPSCK	Filter Prescaler Clock Control. FPSCK controls the prescaling of the clocks used in digital filters for the channel input signals and TCRCLK input, as shown in the table below. Filtering can be controlled independently by each engine, but all input digital filters in the same engine have same clock prescaling. For more details, see Filter clock prescaler . A new value written to FPSCK only becomes effective when the filter prescaler finishes the current count. Table 37-14. Filter prescaler clock control <table><tr><th>Filter control</th><th colspan="2">Sample on system clock divided by:</th></tr><tr><td></td><th>FCSS = 0</th><th>FCSS = 1</th></tr><tr><td>000</td><td>2</td><td>1</td></tr><tr><td>001</td><td>4</td><td>2</td></tr><tr><td>010</td><td>8</td><td>4</td></tr><tr><td>011</td><td>16</td><td>8</td></tr><tr><td>100</td><td>32</td><td>16</td></tr><tr><td>101</td><td>64</td><td>32</td></tr><tr><td>110</td><td>128</td><td>64</td></tr><tr><td>111</td><td>256</td><td>128</td></tr></table>	Filter control	Sample on system clock divided by:			FCSS = 0	FCSS = 1	000	2	1	001	4	2	010	8	4	011	16	8	100	32	16	101	64	32	110	128	64	111	256	128
Filter control	Sample on system clock divided by:																														
	FCSS = 0	FCSS = 1																													
000	2	1																													
001	4	2																													
010	8	4																													
011	16	8																													
100	32	16																													
101	64	32																													
110	128	64																													
111	256	128																													

Table continues on the next page...

eTPU_SCR_ECR_ENG1 field descriptions (continued)

Field	Description										
16–17 CDFC	Channel Digital Filter Control. These bits select a digital filtering mode for the channels when configured as inputs for improved noise immunity (refer to the table below). The eTPU has three digital filtering modes for the channels which provide programmable trade-off between signal latency and noise immunity (see Enhanced Digital Filter - EDF). Changing CDFC during eTPU normal input channel operation is not recommended since it changes the behavior of the transition detection logic while executing its operation. Table 37-13. Channel digital filter control <table> <tr> <th>CDFC</th><th>Selected digital filter</th></tr> <tr> <td>00</td><td>TPU2/3 Two Sample Mode: Using the filter clock which is the system clock divided by (2, 4, 8,..., 256) as a sampling clock (selected by FPSCK), comparing two consecutive samples which agree with each other sets the input signal state. This is the default reset state.</td></tr> <tr> <td>01</td><td>eTPU bypass mode: the input signal is taken unfiltered, also making the channels work on T2/T4 timing mode.¹</td></tr> <tr> <td>10</td><td>eTPU Three Sample Mode: Similar to the TPU2/3 two sample mode, but comparing three consecutive samples which agree with each other sets the input signal state.</td></tr> <tr> <td>11</td><td>eTPU Continuous Mode: Signal need to be stable for the whole filter clock period. This mode compares all the values at the rate of eTPU clock (FCSS=1) or eTPU clock divided by two(FCSS=0), between two consecutive filter clock pulses. Signal needs to be continuously stable for the entire period. If all the values agree with each other, input signal state is updated.</td></tr> </table> 1. See T2/T4 channel timing.	CDFC	Selected digital filter	00	TPU2/3 Two Sample Mode: Using the filter clock which is the system clock divided by (2, 4, 8,..., 256) as a sampling clock (selected by FPSCK), comparing two consecutive samples which agree with each other sets the input signal state. This is the default reset state.	01	eTPU bypass mode: the input signal is taken unfiltered, also making the channels work on T2/T4 timing mode. ¹	10	eTPU Three Sample Mode: Similar to the TPU2/3 two sample mode, but comparing three consecutive samples which agree with each other sets the input signal state.	11	eTPU Continuous Mode: Signal need to be stable for the whole filter clock period. This mode compares all the values at the rate of eTPU clock (FCSS=1) or eTPU clock divided by two(FCSS=0), between two consecutive filter clock pulses. Signal needs to be continuously stable for the entire period. If all the values agree with each other, input signal state is updated.
CDFC	Selected digital filter										
00	TPU2/3 Two Sample Mode: Using the filter clock which is the system clock divided by (2, 4, 8,..., 256) as a sampling clock (selected by FPSCK), comparing two consecutive samples which agree with each other sets the input signal state. This is the default reset state.										
01	eTPU bypass mode: the input signal is taken unfiltered, also making the channels work on T2/T4 timing mode. ¹										
10	eTPU Three Sample Mode: Similar to the TPU2/3 two sample mode, but comparing three consecutive samples which agree with each other sets the input signal state.										
11	eTPU Continuous Mode: Signal need to be stable for the whole filter clock period. This mode compares all the values at the rate of eTPU clock (FCSS=1) or eTPU clock divided by two(FCSS=0), between two consecutive filter clock pulses. Signal needs to be continuously stable for the entire period. If all the values agree with each other, input signal state is updated.										
18 Reserved	Reserved. This field is reserved. This read-only field is reserved and always has the value 0.										
19–23 ERBA	Engine Relative Base Address. This field value is concatenated with the AID instruction field in engine relative address mode to form the SDM address (see Engine Relative Addressing Mode).										
24 SPPDIS	Schedule Priority Passing Disable. SPPDIS is used to disable the priority passing mechanism of the microengine scheduler (see Primary Scheme - Priority Among Channels on Different Levels). NOTE: SPPDIS bit must not be changed while any channel is enabled. 0 Scheduler priority passing mechanism enabled. 1 Scheduler priority passing mechanism disabled.										
25–26 Reserved	Reserved. This field is reserved. This read-only field is reserved and always has the value 0.										
27–31 ETB	Entry Table Base.										

Table continues on the next page...

eTPU_SCR_ECR_ENG1 field descriptions (continued)

Field	Description																					
	The field determines the location of the microcode entry table for the eTPU functions in SCM (see Entry Points). The following table shows the entry table base address options.																					
	Table 37-12. Entry table base address options																					
	<table><tr><th>ETB</th><th>Entry Table Base Address for CPU Host Address (byte format)</th><th>Entry Table Base Address for Microcode Address (word format)</th></tr><tr><td>00000</td><td>0x000</td><td>0x000</td></tr><tr><td>00001</td><td>0x800</td><td>0x200</td></tr><tr><td>00010</td><td>0x1000</td><td>0x400</td></tr><tr><td>-----</td><td>-----</td><td>-----</td></tr><tr><td>11110</td><td>0xF000</td><td>0x3C00</td></tr><tr><td>11111</td><td>0xF800</td><td>0x3E00</td></tr></table>	ETB	Entry Table Base Address for CPU Host Address (byte format)	Entry Table Base Address for Microcode Address (word format)	00000	0x000	0x000	00001	0x800	0x200	00010	0x1000	0x400	-----	-----	-----	11110	0xF000	0x3C00	11111	0xF800	0x3E00
	ETB	Entry Table Base Address for CPU Host Address (byte format)	Entry Table Base Address for Microcode Address (word format)																			
	00000	0x000	0x000																			
	00001	0x800	0x200																			
	00010	0x1000	0x400																			
	-----	-----	-----																			
	11110	0xF000	0x3C00																			
	11111	0xF800	0x3E00																			

3. See [T2/T4 channel timing](#).

37.7.6 eTPU Engine 2 Configuration Register (eTPU_SCR_ECR_ENG2)

This register holds configuration and status fields for engine 2.

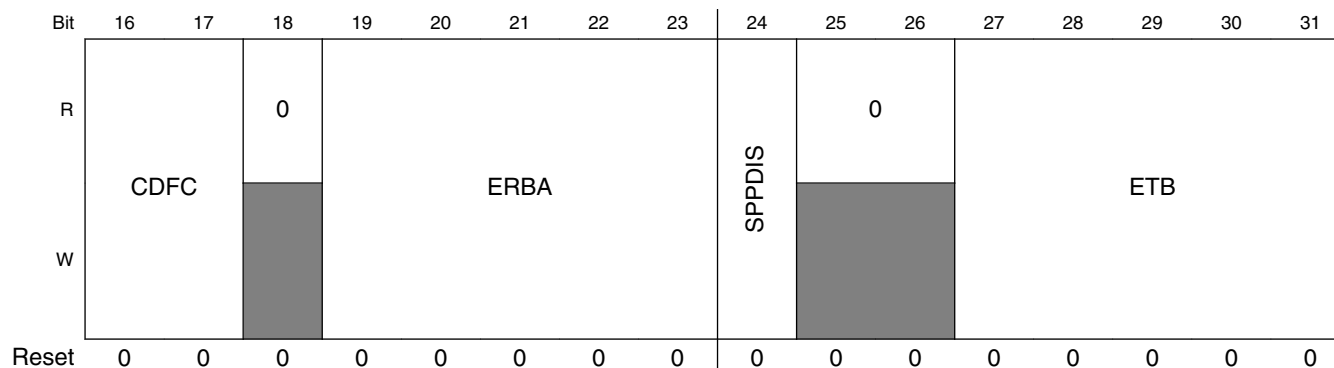
NOTE

See Register Reset Values in the chip-specific eTPU information.

Address: 0h base + 18h offset = 18h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R			0	STF			0		HLTF		0					
W																
Reset	0	*	0	0	0	0	0	0	*	0	0	0	0	0	0	0

System Configuration memory map and registers



* Notes:

- HLTF field: HLTF field reset value is 1 if, after a reset, the eTPU Nexus debug support is on and a breakpoint request is asserted. Otherwise the reset value is 0.
- MDIS field: Reset value is 0.

eTPU_SCR_ECR_ENG2 field descriptions

Field	Description
0 FEND	Force End. FEND assertion terminates any current running thread as if an END instruction have been executed (see Ending Current Thread - END). This bit is self-negating when the thread ends. NOTE: Only on rare occasions (e.g., during a long stall, see Microengine Stall) FEND can be read as 1, because it negates as soon as the end begins execution. Writing FEND = 1 is ignored and FEND stays 0 when the microengine is in TST, halted, stopped, or idle (no thread executing). 0 Normal operation. 1 Ends any ongoing thread.
1 MDIS	Module Disable Bit. When MDIS is set, the Engine shuts down its internal clocks, going into Module Disable Mode. TCR1 and TCR2 cease to increment, and input sampling stops. The Engine asserts the stop flag (STF) bit to indicate that it has stopped. However, the BIU continues to run, and the Host can access all registers except for the channel registers (see channel registers starting with eTPU Channel Configuration-Control registers) NOTE: The Timebase registers can still be read with MDIS=1, but writes are ineffective and a Bus Error is issued. Global Channel Registers and SDM can be accessed normally. After MDIS is set, even before STF asserts, data read from the channel registers is not meaningful and writes are ineffective, issuing a Bus Error. When the MDIS bit is asserted while the microcode is executing, the eTPU will stop when the thread is complete. Stop completes on the next eTPU clock after the stop condition is valid. NOTE: Once MDIS is switched from 1 to 0 or vice versa, it must not write a different value until STF changes accordingly. The MDIS bit is write-protected when VIS=1. See Register Reset Values in the chip-specific eTPU information.

Table continues on the next page...

eTPU_SCR_ECR_ENG2 field descriptions (continued)

Field	Description
	0 eTPU Engine runs. 1 Commands Engine to stop its clocks.
2 Reserved	Reserved. This field is reserved. This read-only field is reserved and always has the value 0.
3 STF	Stop Flag Bit. Each Engine asserts its stop flag (STF) to indicate that it has stopped. Only then the host can assume that the Engine has actually stopped. The eTPU system is fully stopped when the STF bits of both eTPU Engines are asserted. In case of IP-Bus stop, the eTPU system responds by acknowledging stop only after both eTPU 1 and eTPU 2 have been stopped. Engine only stops when any ongoing thread is complete also in this case. Summarizing Engine stop conditions, which STF reflects: STF_1 := (after stop completed) MDIS_1 device stop request. STF_2 := (after stop completed) MDIS_2 device stop request. STF_1 and STF_2 mean STF bit from engine 1 and STF bit from engine 2, respectively. See Register Reset Values in the chip-specific eTPU information. 0 Engine is operating. 1 Engine has stopped (after the local MDIS bit has been asserted, or after the IP-Bus stop line has been asserted).
4–7 Reserved	Reserved. This field is reserved. This read-only field is reserved and always has the value 0.
8 HLTF	Halt Mode Flag. If eTPU Engine entered halt state, this flag is asserted. The flag remains asserted while the microengine is in halt state, even during a single-step or forced instruction execution. See Development support features for further details about entering Halt Mode. 0 eTPU Engine is not halted. 1 eTPU Engine is halted.
9–11 Reserved	Reserved. This field is reserved. This read-only field is reserved and always has the value 0.
12 FCSS	Filter Clock Source Selection Speeds up the filter clock source before the prescaler, allowing more input capture resolution at minimum prescaling. NOTE: FCSS=1 also makes the channel work on T2/T4 timing mode (see T2/T4 channel timing). 0 Use eTPU clock / 2 as EDF clock source before prescaler. 1 Use eTPU clock as EDF clock source before prescaler
13–15 FPSCK	Filter Prescaler Clock Control.

Table continues on the next page...

eTPU_SCR_ECR_ENG2 field descriptions (continued)

Field	Description																		
	FPSCCK controls the prescaling of the clocks used in digital filters for the channel input signals and TCRCLK input, as shown in the table below. Filtering can be controlled independently by each engine, but all input digital filters in the same engine have same clock prescaling. For more details, see Filter clock prescaler. A new value written to FPSCCK only becomes effective when the filter prescaler finishes the current count. Table 37-18. Filter prescaler clock control <table> <tr> <th>Filter control</th><th>Sample on system clock divided by:</th></tr> <tr> <td>000</td><td>2</td></tr> <tr> <td>001</td><td>4</td></tr> <tr> <td>010</td><td>8</td></tr> <tr> <td>011</td><td>16</td></tr> <tr> <td>100</td><td>32</td></tr> <tr> <td>101</td><td>64</td></tr> <tr> <td>110</td><td>128</td></tr> <tr> <td>111</td><td>256</td></tr> </table>	Filter control	Sample on system clock divided by:	000	2	001	4	010	8	011	16	100	32	101	64	110	128	111	256
Filter control	Sample on system clock divided by:																		
000	2																		
001	4																		
010	8																		
011	16																		
100	32																		
101	64																		
110	128																		
111	256																		
16–17 CDFC	Channel Digital Filter Control. These bits select a digital filtering mode for the channels when configured as inputs for improved noise immunity (refer to the table below). The eTPU has three digital filtering modes for the channels which provide programmable trade-off between signal latency and noise immunity (see Enhanced Digital Filter - EDF). Changing CDFC during eTPU normal input channel operation is not recommended since it changes the behavior of the transition detection logic while executing its operation. Table 37-17. Channel digital filter control <table> <tr> <th>CDFC</th><th>Selected digital filter</th></tr> <tr> <td>00</td><td>TPU2/3 Two Sample Mode: Using the filter clock which is the system clock divided by (2, 4, 8,..., 256) as a sampling clock (selected by FPSCCK), comparing two consecutive samples which agree with each other sets the input signal state. This is the default reset state.</td></tr> <tr> <td>01</td><td>eTPU bypass mode: the input signal is taken unfiltered, also making the channels work on T2/T4 timing mode.¹</td></tr> <tr> <td>10</td><td>eTPU Three Sample Mode: Similar to the TPU2/3 two sample mode, but comparing three consecutive samples which agree with each other sets the input signal state.</td></tr> <tr> <td>11</td><td>eTPU Continuous Mode: Signal need to be stable for the whole filter clock period. This mode compares all the values at the rate of eTPU clock (FCSS=1) or eTPU clock divided by two(FCSS=0), between two consecutive filter clock pulses. Signal needs to be continuously stable for the entire period. If all the values agree with each other, input signal state is updated.</td></tr> </table> 1. See T2/T4 channel timing.	CDFC	Selected digital filter	00	TPU2/3 Two Sample Mode: Using the filter clock which is the system clock divided by (2, 4, 8,..., 256) as a sampling clock (selected by FPSCCK), comparing two consecutive samples which agree with each other sets the input signal state. This is the default reset state.	01	eTPU bypass mode: the input signal is taken unfiltered, also making the channels work on T2/T4 timing mode. ¹	10	eTPU Three Sample Mode: Similar to the TPU2/3 two sample mode, but comparing three consecutive samples which agree with each other sets the input signal state.	11	eTPU Continuous Mode: Signal need to be stable for the whole filter clock period. This mode compares all the values at the rate of eTPU clock (FCSS=1) or eTPU clock divided by two(FCSS=0), between two consecutive filter clock pulses. Signal needs to be continuously stable for the entire period. If all the values agree with each other, input signal state is updated.								
CDFC	Selected digital filter																		
00	TPU2/3 Two Sample Mode: Using the filter clock which is the system clock divided by (2, 4, 8,..., 256) as a sampling clock (selected by FPSCCK), comparing two consecutive samples which agree with each other sets the input signal state. This is the default reset state.																		
01	eTPU bypass mode: the input signal is taken unfiltered, also making the channels work on T2/T4 timing mode. ¹																		
10	eTPU Three Sample Mode: Similar to the TPU2/3 two sample mode, but comparing three consecutive samples which agree with each other sets the input signal state.																		
11	eTPU Continuous Mode: Signal need to be stable for the whole filter clock period. This mode compares all the values at the rate of eTPU clock (FCSS=1) or eTPU clock divided by two(FCSS=0), between two consecutive filter clock pulses. Signal needs to be continuously stable for the entire period. If all the values agree with each other, input signal state is updated.																		
18 Reserved	Reserved. This field is reserved. This read-only field is reserved and always has the value 0.																		

Table continues on the next page...

eTPU_SCR_ECR_ENG2 field descriptions (continued)

Field	Description																					
19–23 ERBA	Engine Relative Base Address. This field value is concatenated with the AID instruction field in engine relative address mode to form the SDM address (see Engine Relative Addressing Mode).																					
24 SPPDIS	Schedule Priority Passing Disable. SPPDIS is used to disable the priority passing mechanism of the microengine scheduler (see Primary Scheme - Priority Among Channels on Different Levels). NOTE: SPPDIS bit must not be changed while any channel is enabled. 0 Scheduler priority passing mechanism enabled. 1 Scheduler priority passing mechanism disabled.																					
25–26 Reserved	Reserved. This field is reserved. This read-only field is reserved and always has the value 0.																					
27–31 ETB	Entry Table Base. The field determines the location of the microcode entry table for the eTPU functions in SCM (see Entry Points). The following table shows the entry table base address options. Table 37-16. Entry table base address options <table><tr><th>ETB</th><th>Entry Table Base Address for CPU Host Address (byte format)</th><th>Entry Table Base Address for Microcode Address (word format)</th></tr><tr><td>00000</td><td>0x000</td><td>0x000</td></tr><tr><td>00001</td><td>0x800</td><td>0x200</td></tr><tr><td>00010</td><td>0x1000</td><td>0x400</td></tr><tr><td>-----</td><td>-----</td><td>-----</td></tr><tr><td>11110</td><td>0xF000</td><td>0x3C00</td></tr><tr><td>11111</td><td>0xF800</td><td>0x3E00</td></tr></table>	ETB	Entry Table Base Address for CPU Host Address (byte format)	Entry Table Base Address for Microcode Address (word format)	00000	0x000	0x000	00001	0x800	0x200	00010	0x1000	0x400	-----	-----	-----	11110	0xF000	0x3C00	11111	0xF800	0x3E00
ETB	Entry Table Base Address for CPU Host Address (byte format)	Entry Table Base Address for Microcode Address (word format)																				
00000	0x000	0x000																				
00001	0x800	0x200																				
00010	0x1000	0x400																				
-----	-----	-----																				
11110	0xF000	0x3C00																				
11111	0xF800	0x3E00																				

3. See [T2/T4 channel timing](#).

37.8 Timebase memory map and registers

Time Base registers allows configuration and visibility of internally-generated time bases TCR1 and TCR2. There is one of each of these registers for each eTPU Engine.

NOTE

Writes to these registers issue bus error and are ineffective when MDIS=1. Reads are always allowed.

eTPU_TBR memory map

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
20	eTPU Engine 1 Time Base Configuration Register (eTPU_TBR_TBCR_ENG1)	32	R/W	2000_0000h	37.8.1/1518
24	eTPU Engine 1 Time Base 1 (TCR1) Visibility Register (eTPU_TBR_TB1R_ENG1)	32	R/W	0000_0000h	37.8.2/1522
28	eTPU Engine 1 Time Base 2 (TCR2) Visibility Register (eTPU_TBR_TB2R_ENG1)	32	R/W	0000_0000h	37.8.3/1522
2C	eTPU Engine 1 STAC Configuration Register (eTPU_TBR_STAC_ENG1)	32	R/W	See section	37.8.4/1523
40	eTPU Engine 2 Time Base Configuration Register (eTPU_TBR_TBCR_ENG2)	32	R/W	2000_0000h	37.8.5/1525
44	eTPU Engine 2 Time Base 1 (TCR1) Visibility Register (eTPU_TBR_TB1R_ENG2)	32	R/W	0000_0000h	37.8.6/1529
48	eTPU Engine 2 Time Base 2 (TCR2) Visibility Register (eTPU_TBR_TB2R_ENG2)	32	R/W	0000_0000h	37.8.7/1529
4C	eTPU Engine 2 STAC Configuration Register (eTPU_TBR_STAC_ENG2)	32	R/W	See section	37.8.8/1530

37.8.1 eTPU Engine 1 Time Base Configuration Register (eTPU_TBR_TBCR_ENG1)

This register configures several timebase options.

Address: 0h base + 20h offset = 20h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	TCR2CTL			TCRCF		AM		ATGC	0		TCR2P					
W																
Reset	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	TCR1CTL		TCR1CS	0					TCR1P							
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

eTPU_TBR_TBCR_ENG1 field descriptions

Field	Description
0–2 TCR2CTL	TCR2 Clock/Gate Control.

Table continues on the next page...

eTPU_TBR_TBCR_ENG1 field descriptions (continued)

Field	Description																											
	These bits are part of the TCR2 clocking system (see Time bases). They determine the clock source for TCR2 before the prescaler. TCR2 can count on any detected edge of the TCRCLK signal or use it for gating eTPU clock divided by 8. After reset - TCRCLK signal rising edge is selected. TCR2 can also be clocked by the eTPU clock divided by 8. TCR2CTL also determines the TCRCLK edge selected for angle tooth detection in angle mode. See the following table. Table 37-24. TCR2 Clock Source <table><tr><th>TCR2CTL</th><th>AM = 0 TCR2 Clock before prescaler</th><th>AM > 0 Angle Tooth detection</th></tr><tr><td>000</td><td>Gated DIV8 clock (eTPU clock / 8). In this case, when the external TCRCLK signal is low, the DIV8 clock is blocked, preventing it from incrementing the TCR2 prescaler. When the external TCRCLK signal is high, TCR2 prescaler is incremented at the frequency of the eTPU clock divided by 8.</td><td>Do not use with AM > 0</td></tr><tr><td>001</td><td>Rise transition on TCRCLK signal increments the TCR2 prescaler.</td><td>Rising edge</td></tr><tr><td>010</td><td>Fall transition on TCRCLK signal increments the TCR2 prescaler.</td><td>Falling edge</td></tr><tr><td>011</td><td>Rise or Fall transition on TCRCLK signal increments the TCR2 prescaler.</td><td>Both edges</td></tr><tr><td>100</td><td>DIV8 clock (eTPU clock / 8)</td><td>Do not use with AM > 0</td></tr><tr><td>101</td><td>Do not use with AM=0</td><td>Do not use with AM > 0</td></tr><tr><td>110</td><td>Do not use with AM=0</td><td>No edge¹</td></tr><tr><td>111</td><td>TCR2 frozen, except as STAC client</td><td>Do not use with AM > 0</td></tr></table> 1. TCRCLK edges are not detected by the EAC logic, but they can still be detected by the channel 0 logic if AM=01.	TCR2CTL	AM = 0 TCR2 Clock before prescaler	AM > 0 Angle Tooth detection	000	Gated DIV8 clock (eTPU clock / 8). In this case, when the external TCRCLK signal is low, the DIV8 clock is blocked, preventing it from incrementing the TCR2 prescaler. When the external TCRCLK signal is high, TCR2 prescaler is incremented at the frequency of the eTPU clock divided by 8.	Do not use with AM > 0	001	Rise transition on TCRCLK signal increments the TCR2 prescaler.	Rising edge	010	Fall transition on TCRCLK signal increments the TCR2 prescaler.	Falling edge	011	Rise or Fall transition on TCRCLK signal increments the TCR2 prescaler.	Both edges	100	DIV8 clock (eTPU clock / 8)	Do not use with AM > 0	101	Do not use with AM=0	Do not use with AM > 0	110	Do not use with AM=0	No edge ¹	111	TCR2 frozen, except as STAC client	Do not use with AM > 0
TCR2CTL	AM = 0 TCR2 Clock before prescaler	AM > 0 Angle Tooth detection																										
000	Gated DIV8 clock (eTPU clock / 8). In this case, when the external TCRCLK signal is low, the DIV8 clock is blocked, preventing it from incrementing the TCR2 prescaler. When the external TCRCLK signal is high, TCR2 prescaler is incremented at the frequency of the eTPU clock divided by 8.	Do not use with AM > 0																										
001	Rise transition on TCRCLK signal increments the TCR2 prescaler.	Rising edge																										
010	Fall transition on TCRCLK signal increments the TCR2 prescaler.	Falling edge																										
011	Rise or Fall transition on TCRCLK signal increments the TCR2 prescaler.	Both edges																										
100	DIV8 clock (eTPU clock / 8)	Do not use with AM > 0																										
101	Do not use with AM=0	Do not use with AM > 0																										
110	Do not use with AM=0	No edge ¹																										
111	TCR2 frozen, except as STAC client	Do not use with AM > 0																										
3–4 TCRCF	TCRCLK Signal Filter Control. This field controls the TCRCLK digital filter (see TCRCLK digital filter), determining whether the TCRCLK signal input (after a synchronizer) is filtered with the same filter clock as the channel input signals (see Enhanced Digital Filter - EDF) or uses the eTPU clock divided by 2, and also whether the TCRCLK digital filter works in integrator mode or two sample mode (see the following table). Table 37-23. TCRCLK Filter Clock/Mode <table><tr><th>TCRCF</th><th>Filter Clock</th><th>Filter Mode</th></tr><tr><td>00</td><td>eTPU clock divided by 2</td><td>Two sample</td></tr><tr><td>01</td><td>Filter clock of the channels</td><td>Two sample</td></tr><tr><td>10</td><td>eTPU clock divided by 2</td><td>Integrator</td></tr><tr><td>11</td><td>Filter clock of the channels</td><td>Integrator</td></tr></table>	TCRCF	Filter Clock	Filter Mode	00	eTPU clock divided by 2	Two sample	01	Filter clock of the channels	Two sample	10	eTPU clock divided by 2	Integrator	11	Filter clock of the channels	Integrator												
TCRCF	Filter Clock	Filter Mode																										
00	eTPU clock divided by 2	Two sample																										
01	Filter clock of the channels	Two sample																										
10	eTPU clock divided by 2	Integrator																										
11	Filter clock of the channels	Integrator																										
5–6 AM	Angle Mode Selection. This field enables the Enhanced Angle Counter logic to generate angle information (see EAC - eTPU Angle Counter), and also select the tooth signal input and the channel used to process it, as shown in the following table. When EAC is not disabled by AM and neither TCR1 nor TCR2 are STAC Clients, the EAC																											

Table continues on the next page...

eTPU_TBR_TBCR_ENG1 field descriptions (continued)

Field	Description																		
	(eTPU Angle Clock) hardware provides angle information to the channels using the TCR2 bus. When AM is reset (non-angle mode), the EAC operation is disabled, and its internal registers can be used as general purpose. For more information, see EAC - eTPU Angle Counter. Table 37-22. AM - Angle Mode Selection <table><tr><th>Value</th><th>TCR2 Value</th><th>Tooth signal</th><th>Tooth processing channel</th></tr><tr><td>00</td><td>Timebase (EAC operation disabled)</td><td colspan="2">Not applicable</td></tr><tr><td>01</td><td rowspan="3">Angle Ticks</td><td>TCRCLK input</td><td>0</td></tr><tr><td>10</td><td>Channel 1 input</td><td>1</td></tr><tr><td>11</td><td>Channel 2 input</td><td>2</td></tr></table> If TCR1 or TCR2 is a STAC Bus Client (see STAC interface), the EAC operation is not allowed, and AM must be set to 0. NOTE: Changing AM may cause unexpected transition detections on the channel selected by AM, depending on the channel mode and state (see Transition detection and Time Base Capture). If AM must be changed with GTBE=1, the recommended procedure is described in Restarting angle logic.	Value	TCR2 Value	Tooth signal	Tooth processing channel	00	Timebase (EAC operation disabled)	Not applicable		01	Angle Ticks	TCRCLK input	0	10	Channel 1 input	1	11	Channel 2 input	2
Value	TCR2 Value	Tooth signal	Tooth processing channel																
00	Timebase (EAC operation disabled)	Not applicable																	
01	Angle Ticks	TCRCLK input	0																
10		Channel 1 input	1																
11		Channel 2 input	2																
7 ATGC	Angle Tick Generator Clock. This bit selects the clock that provides the time interval unit for Angle Tick Generation. For details, see Generating the angle ticks. 0 TCR1 prescaler output provides time intervals for tick generation. 1 TCR2 prescaler output provides time intervals for tick generation.																		
8–9 Reserved	Reserved. This field is reserved. This read-only field is reserved and always has the value 0.																		
10–15 TCR2P	Timer Count Register 2 Prescaler Control. These bits are part of the TCR2 clocking system (see Time bases). TCR2 is clocked from the output of a prescaler. The prescaler divides its input by (TCR2P + 1) allowing frequency divisions from 1 to 64. The prescaler input is the eTPU clock divided by 8 (in gated or non-gated clock mode), or Internal Timebase input, or TCRCLK filtered input. This field has no effect on TCR2 in Angle Mode.																		
16–17 TCR1CTL	TCR1 Clock/Gate Control. TCR1CTL is part of the TCR1 clocking system (see Time bases). It determines, together with TCR1CS, the clock source for TCR1. TCR1 can count on detected rising edge of the TCRCLK signal, the eTPU clock, or the eTPU clock divided by 2 (see the table in the TCR1CS bit description). After reset TCRCLK signal is selected.																		
18 TCR1CS	TCR1 Clock Source TCR1CS provides the option to double the TCR1 incrementing speed, using eTPU clock as its clock source instead of eTPU clock / 2.																		

Table continues on the next page...

eTPU_TBR_TBCR_ENG1 field descriptions (continued)

Field	Description																					
	NOTE: If ATGC=0, the clock source of the EAC angle tick generator will still be an even division of eTPU clock if TCR1CS=1, obeying to the field TCR1P as if TCR1CS=0 (see Angle Tick Generator). Table 37-21. TCR1 Clock Source <table><tr><th>TCR1CTL</th><th>TCR1CS</th><th>TCR1 Clock before prescaler</th></tr><tr><td>00</td><td>0</td><td>Selects TCRCLK as clock source for the TCR1 prescaler.¹</td></tr><tr><td>01</td><td>0</td><td>Reserved</td></tr><tr><td>01</td><td>1</td><td>Reserved</td></tr><tr><td>10</td><td>0</td><td>Selects eTPU clock divided by 2 as clock source for the TCR1 prescaler.</td></tr><tr><td>10</td><td>1</td><td>Selects eTPU clock as clock source for the TCR1 prescaler.</td></tr><tr><td>11</td><td>0</td><td>TCR1 frozen, except as a STAC client</td></tr></table> 1. This selection must not be used in Angle Mode. 0 Use eTPU clock / 2 as TCR1 clock source before the prescaler, if that clock source is selected by TCR1CTL. 1 Use eTPU clock as TCR1 clock source before the prescaler; can only be set in specific combinations with TCR1CTL, as shown in the table. NOTE: TCR1CS=1 also makes the channel work on T2/T4 timing mode (see T2/T4 channel timing).	TCR1CTL	TCR1CS	TCR1 Clock before prescaler	00	0	Selects TCRCLK as clock source for the TCR1 prescaler. ¹	01	0	Reserved	01	1	Reserved	10	0	Selects eTPU clock divided by 2 as clock source for the TCR1 prescaler.	10	1	Selects eTPU clock as clock source for the TCR1 prescaler.	11	0	TCR1 frozen, except as a STAC client
TCR1CTL	TCR1CS	TCR1 Clock before prescaler																				
00	0	Selects TCRCLK as clock source for the TCR1 prescaler. ¹																				
01	0	Reserved																				
01	1	Reserved																				
10	0	Selects eTPU clock divided by 2 as clock source for the TCR1 prescaler.																				
10	1	Selects eTPU clock as clock source for the TCR1 prescaler.																				
11	0	TCR1 frozen, except as a STAC client																				
19–23 Reserved	Reserved. This field is reserved. This read-only field is reserved and always has the value 0.																					
24–31 TCR1P	Timer Count Register 1 Prescaler Control. TCR1 is clocked from the output of a prescaler. The input to the prescaler is the internal eTPU clock divided by 2, eTPU clock, or the output of TCRCLK filter. The prescaler divides this input by (TCR1P + 1) allowing frequency divisions from 1 up to 256.																					

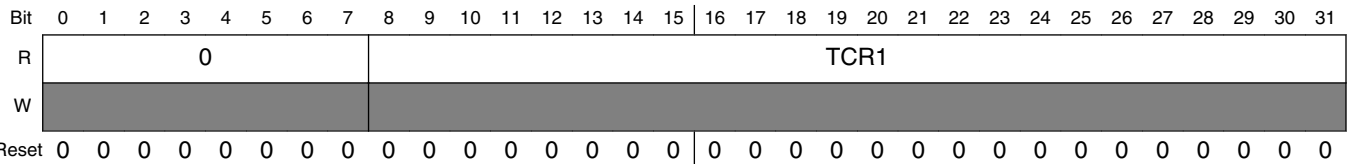
1. TCRCLK edges are not detected by the EAC logic, but they can still be detected by the channel 0 logic if AM=01.

2. This selection must not be used in Angle Mode.

37.8.2 eTPU Engine 1 Time Base 1 (TCR1) Visibility Register (eTPU_TBR_TB1R_ENG1)

This register provides visibility of the TCR1 time base for host read access (see [Time bases](#)). This register is read-only. The value of the TCR1 time base shown can be driven by the TCR1 counter or imported from STAC bus, depending on the configuration set in [eTPU Engine 1 STAC Configuration Register \(eTPU_TBR_STAC_ENG1\)](#).

Address: 0h base + 24h offset = 24h



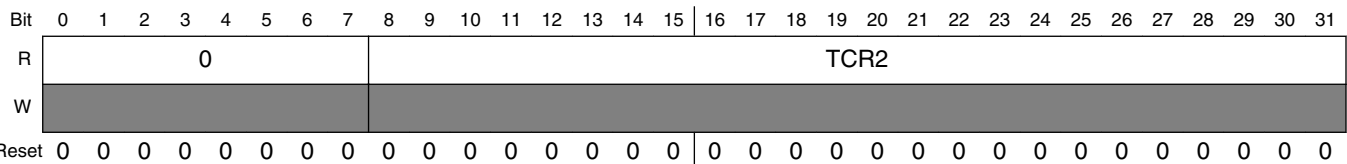
eTPU_TBR_TB1R_ENG1 field descriptions

Field	Description
0–7 Reserved	Reserved. This field is reserved. This read-only field is reserved and always has the value 0.
8–31 TCR1	TCR1 value. TCR1 value used on matches and captures. See Time bases .

37.8.3 eTPU Engine 1 Time Base 2 (TCR2) Visibility Register (eTPU_TBR_TB2R_ENG1)

This register provides visibility of the TCR2 time base for host read access (see [Time bases](#)). This register is read-only. The value of the TCR2 time base shown can be driven by the TCR2 counter or imported from the STAC bus, depending on Angle Mode and STAC configurations set in [eTPU Engine 1 STAC Configuration Register \(eTPU_TBR_STAC_ENG1\)](#).

Address: 0h base + 28h offset = 28h



eTPU_TBR_TB2R_ENG1 field descriptions

Field	Description
0–7 Reserved	Reserved. This field is reserved. This read-only field is reserved and always has the value 0.
8–31 TCR2	TCR2 value. TCR2 value used on matches and captures. See Time bases .

37.8.4 eTPU Engine 1 STAC Configuration Register (eTPU_TBR_STAC_ENG1)

This register configures the eTPU STAC bus operation as a STAC Server/Client module (see [STAC interface](#)).

Address: 0h base + 2Ch offset = 2Ch

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	REN1	RSC1	0	SERVER_ID1					0				SRV1			
W																
Reset	0	0	0	0	*	*	*	*	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	REN2	RSC2	0	SERVER_ID2					0				SRV2			
W																
Reset	0	0	0	0	*	*	*	*	0	0	0	0	0	0	0	0

* Notes:

- SERVER_ID2 field: See the chip-specific eTPU information for the reset value.
- SERVER_ID1 field: See the chip-specific eTPU information for the reset value.

eTPU_TBR_STAC_ENG1 field descriptions

Field	Description
0 REN1	TCR1 Resource Client/Server Operation Enable Bit. This bit enables Client/Server operation to eTPU STAC resources. REN1 enables TCR1 STAC bus operations. 0 Server/Client Operation for TCR1 is disabled. 1 Server/Client Operation for TCR1 is enabled.
1 RSC1	TCR1 Resource Server/Client Assignment Bits. This bit selects the eTPU data resource assignment to be used as Servers or Clients. For Server mode, MCU-dependent value determines the unique server address assigned to each TCR. See the chip-specific

Table continues on the next page...

eTPU_TBR_STAC_ENG1 field descriptions (continued)

Field	Description
	eTPU information for the STAC server IDs. For a Client mode, the SRV1 field determines the Server address to which the Client listens. NOTE: When TCR1 or TCR2 is configured as a STAC Bus Client (REN1/2 = 1, RSC1/2 = 0) the eTPU Angle Clock hardware cannot be used. NOTE: RSC1 must not be changed when the REN1 bit is asserted. 0 Resource Client operation. 1 Resource Server operation.
2–3 Reserved	Reserved. This field is reserved. This read-only field is reserved and always has the value 0.
4–7 SERVER_ID1	STAC ID 1. STAC Server ID (read-only MCU-dependent values; see the chip-specific eTPU information for the STAC server IDs) used for TCR1 when STAC servers are used.
8–11 Reserved	Reserved. This field is reserved. This read-only field is reserved and always has the value 0.
12–15 SRV1	TCR1 Resource Server. This bit selects the address of the specific STAC Server to which the local TCR1 listens when configured as a STAC Client. SRV1 selects the STAC Server of TCR1.
16 REN2	TCR2 Resource Client/Server Operation Enable Bit. This bit enables Client/Server operation to eTPU STAC resources. REN2 enables TCR2 STAC bus operations. 0 Server/Client Operation for TCR2 is disabled. 1 Server/Client Operation for TCR2 is enabled.
17 RSC2	TCR2 Resource Server/Client Assignment Bits. NOTE: Either time or angle TCR2 values. 0 Resource Client operation. 1 Resource Server operation.
18–19 Reserved	Reserved. This field is reserved. This read-only field is reserved and always has the value 0.
20–23 SERVER_ID2	STAC ID 2. STAC Server ID (read-only MCU-dependent values; see the chip-specific eTPU information for the STAC server IDs) used for TCR2 when when STAC servers are used.
24–27 Reserved	Reserved. This field is reserved. This read-only field is reserved and always has the value 0.

Table continues on the next page...

eTPU_TBR_STAC_ENG1 field descriptions (continued)

Field	Description
28–31 SRV2	TCR2 Resource Server. This bit selects the address of the specific STAC Server to which the local TCR2 listens when configured as a STAC Client. SRV2 selects the Red Line STAC Server of TCR2.

37.8.5 eTPU Engine 2 Time Base Configuration Register (eTPU_TBR_TBCR_ENG2)

This register configures several timebase options.

Address: 0h base + 40h offset = 40h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	TCR2CTL		TCRCF		AM		ATGC		0		TCR2P					
W																
Reset	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0
Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	TCR1CTL		TCR1CS	0					TCR1P							
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

eTPU_TBR_TBCR_ENG2 field descriptions

Field	Description
0–2 TCR2CTL	TCR2 Clock/Gate Control. These bits are part of the TCR2 clocking system (see Time bases). They determine the clock source for TCR2 before the prescaler. TCR2 can count on any detected edge of the TCRCLK signal or use it for gating eTPU clock divided by 8. After reset - TCRCLK signal rising edge is selected. TCR2 can also be

Table continues on the next page...

eTPU_TBR_TBCR_ENG2 field descriptions (continued)

Field	Description																											
	clocked by the eTPU clock divided by 8. TCR2CTL also determines the TCRCLK edge selected for angle tooth detection in angle mode. See the following table. Table 37-32. TCR2 Clock Source <table><tr><th>TCR2CTL</th><th>AM = 0 TCR2 Clock before prescaler</th><th>AM > 0 Angle Tooth detection</th></tr><tr><td>000</td><td>Gated DIV8 clock (eTPU clock / 8). In this case, when the external TCRCLK signal is low, the DIV8 clock is blocked, preventing it from incrementing the TCR2 prescaler. When the external TCRCLK signal is high, TCR2 prescaler is incremented at the frequency of the eTPU clock divided by 8.</td><td>Do not use with AM > 0</td></tr><tr><td>001</td><td>Rise transition on TCRCLK signal increments the TCR2 prescaler.</td><td>Rising edge</td></tr><tr><td>010</td><td>Fall transition on TCRCLK signal increments the TCR2 prescaler.</td><td>Falling edge</td></tr><tr><td>011</td><td>Rise or Fall transition on TCRCLK signal increments the TCR2 prescaler.</td><td>Both edges</td></tr><tr><td>100</td><td>DIV8 clock (eTPU clock / 8)</td><td>Do not use with AM > 0</td></tr><tr><td>101</td><td>Do not use with AM=0</td><td>Do not use with AM > 0</td></tr><tr><td>110</td><td>Do not use with AM=0</td><td>No edge¹</td></tr><tr><td>111</td><td>TCR2 frozen, except as STAC client</td><td>Do not use with AM > 0</td></tr></table> 1. TCRCLK edges are not detected by the EAC logic, but they can still be detected by the channel 0 logic if AM=01.	TCR2CTL	AM = 0 TCR2 Clock before prescaler	AM > 0 Angle Tooth detection	000	Gated DIV8 clock (eTPU clock / 8). In this case, when the external TCRCLK signal is low, the DIV8 clock is blocked, preventing it from incrementing the TCR2 prescaler. When the external TCRCLK signal is high, TCR2 prescaler is incremented at the frequency of the eTPU clock divided by 8.	Do not use with AM > 0	001	Rise transition on TCRCLK signal increments the TCR2 prescaler.	Rising edge	010	Fall transition on TCRCLK signal increments the TCR2 prescaler.	Falling edge	011	Rise or Fall transition on TCRCLK signal increments the TCR2 prescaler.	Both edges	100	DIV8 clock (eTPU clock / 8)	Do not use with AM > 0	101	Do not use with AM=0	Do not use with AM > 0	110	Do not use with AM=0	No edge ¹	111	TCR2 frozen, except as STAC client	Do not use with AM > 0
TCR2CTL	AM = 0 TCR2 Clock before prescaler	AM > 0 Angle Tooth detection																										
000	Gated DIV8 clock (eTPU clock / 8). In this case, when the external TCRCLK signal is low, the DIV8 clock is blocked, preventing it from incrementing the TCR2 prescaler. When the external TCRCLK signal is high, TCR2 prescaler is incremented at the frequency of the eTPU clock divided by 8.	Do not use with AM > 0																										
001	Rise transition on TCRCLK signal increments the TCR2 prescaler.	Rising edge																										
010	Fall transition on TCRCLK signal increments the TCR2 prescaler.	Falling edge																										
011	Rise or Fall transition on TCRCLK signal increments the TCR2 prescaler.	Both edges																										
100	DIV8 clock (eTPU clock / 8)	Do not use with AM > 0																										
101	Do not use with AM=0	Do not use with AM > 0																										
110	Do not use with AM=0	No edge ¹																										
111	TCR2 frozen, except as STAC client	Do not use with AM > 0																										
3–4 TCRCF	TCRCLK Signal Filter Control. This field controls the TCRCLK digital filter (see TCRCLK digital filter, determining whether the TCRCLK signal input (after a synchronizer) is filtered with the same filter clock as the channel input signals (see Enhanced Digital Filter - EDF) or uses the eTPU clock divided by 2, and also whether the TCRCLK digital filter works in integrator mode or two sample mode (see the following table). Table 37-31. TCRCLK Filter Clock/Mode <table><tr><th>TCRCF</th><th>Filter Clock</th><th>Filter Mode</th></tr><tr><td>00</td><td>eTPU clock divided by 2</td><td>Two sample</td></tr><tr><td>01</td><td>Filter clock of the channels</td><td>Two sample</td></tr><tr><td>10</td><td>eTPU clock divided by 2</td><td>Integrator</td></tr><tr><td>11</td><td>Filter clock of the channels</td><td>Integrator</td></tr></table>	TCRCF	Filter Clock	Filter Mode	00	eTPU clock divided by 2	Two sample	01	Filter clock of the channels	Two sample	10	eTPU clock divided by 2	Integrator	11	Filter clock of the channels	Integrator												
TCRCF	Filter Clock	Filter Mode																										
00	eTPU clock divided by 2	Two sample																										
01	Filter clock of the channels	Two sample																										
10	eTPU clock divided by 2	Integrator																										
11	Filter clock of the channels	Integrator																										
5–6 AM	Angle Mode Selection. This field enables the Enhanced Angle Counter logic to generate angle information (see EAC - eTPU Angle Counter), and also select the tooth signal input and the channel used to process it, as shown in the following table. When EAC is not disabled by AM and neither TCR1 nor TCR2 are STAC Clients, the EAC (eTPU Angle Clock) hardware provides angle information to the channels using the TCR2 bus. When AM																											

Table continues on the next page...

eTPU_TBR_TBCR_ENG2 field descriptions (continued)

Field	Description																		
	is reset (non-angle mode), the EAC operation is disabled, and its internal registers can be used as general purpose. For more information, see EAC - eTPU Angle Counter. Table 37-30. AM - Angle Mode Selection <table><tr><th>Value</th><th>TCR2 Value</th><th>Tooth signal</th><th>Tooth processing channel</th></tr><tr><td>00</td><td>Timebase (EAC operation disabled)</td><td colspan="2">Not applicable</td></tr><tr><td>01</td><td rowspan="3">Angle Ticks</td><td>TCRCLK input</td><td>0</td></tr><tr><td>10</td><td>Channel 1 input</td><td>1</td></tr><tr><td>11</td><td>Channel 2 input</td><td>2</td></tr></table> If TCR1 or TCR2 is a STAC Bus Client (see STAC interface), the EAC operation is not allowed, and AM must be set to 0. NOTE: Changing AM may cause unexpected transition detections on the channel selected by AM, depending on the channel mode and state (see Transition detection and Time Base Capture). If AM must be changed with GTBE=1, the recommended procedure is described in Restarting angle logic.	Value	TCR2 Value	Tooth signal	Tooth processing channel	00	Timebase (EAC operation disabled)	Not applicable		01	Angle Ticks	TCRCLK input	0	10	Channel 1 input	1	11	Channel 2 input	2
Value	TCR2 Value	Tooth signal	Tooth processing channel																
00	Timebase (EAC operation disabled)	Not applicable																	
01	Angle Ticks	TCRCLK input	0																
10		Channel 1 input	1																
11		Channel 2 input	2																
7 ATGC	Angle Tick Generator Clock. This bit selects the clock that provides the time interval unit for Angle Tick Generation. For details, see Generating the angle ticks. 0 TCR1 prescaler output provides time intervals for tick generation. 1 TCR2 prescaler output provides time intervals for tick generation.																		
8–9 Reserved	Reserved. This field is reserved. This read-only field is reserved and always has the value 0.																		
10–15 TCR2P	Timer Count Register 2 Prescaler Control. These bits are part of the TCR2 clocking system (see Time bases). TCR2 is clocked from the output of a prescaler. The prescaler divides its input by (TCR2P + 1) allowing frequency divisions from 1 to 64. The prescaler input is the eTPU clock divided by 8 (in gated or non-gated clock mode), or Internal Timebase input, or TCRCLK filtered input. This field has no effect on TCR2 in Angle Mode.																		
16–17 TCR1CTL	TCR1 Clock/Gate Control. TCR1CTL is part of the TCR1 clocking system (see Time bases). It determines, together with TCR1CS, the clock source for TCR1. TCR1 can count on detected rising edge of the TCRCLK signal, the eTPU clock, or the eTPU clock divided by 2 (see the table in the TCR1CS bit description). After reset TCRCLK signal is selected.																		
18 TCR1CS	TCR1 Clock Source TCR1CS provides the option to double the TCR1 incrementing speed, using eTPU clock as its clock source instead of eTPU clock / 2.																		

Table continues on the next page...

eTPU_TBR_TBCR_ENG2 field descriptions (continued)

Field	Description																					
	NOTE: If ATGC=0, the clock source of the EAC angle tick generator will still be an even division of eTPU clock if TCR1CS=1, obeying to the field TCR1P as if TCR1CS=0 (see Angle Tick Generator). Table 37-29. TCR1 Clock Source <table><tr><th>TCR1CTL</th><th>TCR1CS</th><th>TCR1 Clock before prescaler</th></tr><tr><td>00</td><td>0</td><td>Selects TCRCLK as clock source for the TCR1 prescaler.¹</td></tr><tr><td>01</td><td>0</td><td>Reserved</td></tr><tr><td>01</td><td>1</td><td>Reserved</td></tr><tr><td>10</td><td>0</td><td>Selects eTPU clock divided by 2 as clock source for the TCR1 prescaler.</td></tr><tr><td>10</td><td>1</td><td>Selects eTPU clock as clock source for the TCR1 prescaler.</td></tr><tr><td>11</td><td>0</td><td>TCR1 frozen, except as a STAC client</td></tr></table> 1. This selection must not be used in Angle Mode. 0 Use eTPU clock / 2 as TCR1 clock source before the prescaler, if that clock source is selected by TCR1CTL. 1 Use eTPU clock as TCR1 clock source before the prescaler; can only be set in specific combinations with TCR1CTL, as shown in the table. NOTE: TCR1CS=1 also makes the channel work on T2/T4 timing mode (see T2/T4 channel timing).	TCR1CTL	TCR1CS	TCR1 Clock before prescaler	00	0	Selects TCRCLK as clock source for the TCR1 prescaler. ¹	01	0	Reserved	01	1	Reserved	10	0	Selects eTPU clock divided by 2 as clock source for the TCR1 prescaler.	10	1	Selects eTPU clock as clock source for the TCR1 prescaler.	11	0	TCR1 frozen, except as a STAC client
TCR1CTL	TCR1CS	TCR1 Clock before prescaler																				
00	0	Selects TCRCLK as clock source for the TCR1 prescaler. ¹																				
01	0	Reserved																				
01	1	Reserved																				
10	0	Selects eTPU clock divided by 2 as clock source for the TCR1 prescaler.																				
10	1	Selects eTPU clock as clock source for the TCR1 prescaler.																				
11	0	TCR1 frozen, except as a STAC client																				
19–23 Reserved	Reserved. This field is reserved. This read-only field is reserved and always has the value 0.																					
24–31 TCR1P	Timer Count Register 1 Prescaler Control. TCR1 is clocked from the output of a prescaler. The input to the prescaler is the internal eTPU clock divided by 2, eTPU clock, or the output of TCRCLK filter. The prescaler divides this input by (TCR1P + 1) allowing frequency divisions from 1 up to 256.																					

1. TCRCLK edges are not detected by the EAC logic, but they can still be detected by the channel 0 logic if AM=01.

2. This selection must not be used in Angle Mode.

37.8.6 eTPU Engine 2 Time Base 1 (TCR1) Visibility Register (eTPU_TBR_TB1R_ENG2)

This register provides visibility of the TCR1 time base for host read access (see [Time bases](#)). This register is read-only. The value of the TCR1 time base shown can be driven by the TCR1 counter or imported from STAC bus, depending on the configuration set in [eTPU Engine 2 STAC Configuration Register \(eTPU_TBR_STAC_ENG2\)](#).

Address: 0h base + 44h offset = 44h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0								TCR1																							
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	

eTPU_TBR_TB1R_ENG2 field descriptions

Field	Description
0–7 Reserved	Reserved. This field is reserved. This read-only field is reserved and always has the value 0.
8–31 TCR1	TCR1 value. TCR1 value used on matches and captures. See Time bases .

37.8.7 eTPU Engine 2 Time Base 2 (TCR2) Visibility Register (eTPU_TBR_TB2R_ENG2)

This register provides visibility of the TCR2 time base for host read access (see [Time bases](#)). This register is read-only. The value of the TCR2 time base shown can be driven by the TCR2 counter or imported from the STAC bus, depending on Angle Mode and STAC configurations set in [eTPU Engine 2 STAC Configuration Register \(eTPU_TBR_STAC_ENG2\)](#).

Address: 0h base + 48h offset = 48h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0								TCR2																							
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	

eTPU_TBR_TB2R_ENG2 field descriptions

Field	Description
0–7 Reserved	Reserved. This field is reserved. This read-only field is reserved and always has the value 0.
8–31 TCR2	TCR2 value. TCR2 value used on matches and captures. See Time bases .

37.8.8 eTPU Engine 2 STAC Configuration Register (eTPU_TBR_STAC_ENG2)

This register configures the eTPU STAC bus operation as a STAC Server/Client module (see [STAC interface](#)).

Address: 0h base + 4Ch offset = 4Ch

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R			0		SERVER_ID1						0					
W	REN1	RSC1													SRV1	
Reset	0	0	0	0	*	*	*	*	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R			0		SERVER_ID2						0					
W	REN2	RSC2													SRV2	
Reset	0	0	0	0	*	*	*	*	0	0	0	0	0	0	0	0

* Notes:

- SERVER_ID2 field: See the chip-specific eTPU information for the reset value.
- SERVER_ID1 field: See the chip-specific eTPU information for the reset value.

eTPU_TBR_STAC_ENG2 field descriptions

Field	Description
0 REN1	TCR1 Resource Client/Server Operation Enable Bit. This bit enables Client/Server operation to eTPU STAC resources. REN1 enables TCR1 STAC bus operations. 0 Server/Client Operation for TCR1 is disabled. 1 Server/Client Operation for TCR1 is enabled.
1 RSC1	TCR1 Resource Server/Client Assignment Bits. This bit selects the eTPU data resource assignment to be used as Servers or Clients. For Server mode, MCU-dependent value determines the unique server address assigned to each TCR. See the chip-specific

Table continues on the next page...

eTPU_TBR_STAC_ENG2 field descriptions (continued)

Field	Description
	eTPU information for the STAC server IDs. For a Client mode, the SRV1 field determines the Server address to which the Client listens. NOTE: Resource identifies any parameter that changes along the time and can be exported / imported from other device. In eTPU context, a resource can be TCR1 or TCR2 (either Time or Angle values). NOTE: When TCR1 or TCR2 is configured as a STAC Bus Client (REN2 = 1, RSC2 = 0) the eTPU Angle Clock hardware cannot be used. NOTE: RSC1 must not be changed when the REN1 bit is asserted. 0 TCR1 Client operation. 1 TCR1 Server operation.
2–3 Reserved	Reserved. This field is reserved. This read-only field is reserved and always has the value 0.
4–7 SERVER_ID1	STAC ID 1. STAC Server ID (read-only MCU-dependent values; see the chip-specific eTPU information for the STAC server IDs) used for TCR1 when STAC servers are used.
8–11 Reserved	Reserved. This field is reserved. This read-only field is reserved and always has the value 0.
12–15 SRV1	TCR1 Resource Server. This bit selects the address of the specific STAC Server to which the local TCR1 listens when configured as a STAC Client. SRV1 selects the STAC Server of TCR1.
16 REN2	TCR2 Resource Client/Server Operation Enable Bit. This bit enables Client/Server operation to eTPU STAC resources. REN2 enables TCR2 STAC bus operations. 0 Server/Client Operation for resource 2 is disabled. 1 Server/Client Operation for resource 2 is enabled.
17 RSC2	TCR2 Resource Server/Client Assignment Bits. NOTE: Resource identifies any parameter that changes along the time and can be exported / imported from other device. In eTPU context, a resource can be TCR1 or TCR2 (either Time or Angle values). 0 Resource Client operation. 1 Resource Server operation.
18–19 Reserved	Reserved. This field is reserved. This read-only field is reserved and always has the value 0.
20–23 SERVER_ID2	STAC ID 2.

Table continues on the next page...

eTPU_TBR_STAC_ENG2 field descriptions (continued)

Field	Description
	STAC Server ID (read-only MCU-dependent values; see the chip-specific eTPU information for the STAC server IDs) used for TCR2 when STAC servers are used.
24–27 Reserved	Reserved. This field is reserved. This read-only field is reserved and always has the value 0.
28–31 SRV2	TCR2 Resource Server. This bit selects the address of the specific STAC Server to which the local TCR2 listens when configured as a STAC Client. SRV2 selects the Red Line STAC Server of TCR2.

37.9 Engine memory map and registers

This section gathers registers that are engine-related, other than [eTPU Engine 1 Configuration Register \(eTPU_SCR_ECR_ENG1\)](#) and [eTPU Engine 2 Configuration Register \(eTPU_SCR_ECR_ENG2\)](#).

eTPU_ER memory map

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
60	eTPU Watchdog Timer Register for Engine 1 (eTPU_ER_WDTR_ENG1)	32	R/W	0000_0000h	37.9.1/1532
68	eTPU Idle Register for Engine 1 (eTPU_ER_IDLER_ENG1)	32	R/W	Undefined	37.9.2/1534
70	eTPU Watchdog Timer Register for Engine 2 (eTPU_ER_WDTR_ENG2)	32	R/W	0000_0000h	37.9.3/1535
78	eTPU Idle Register for Engine 2 (eTPU_ER_IDLER_ENG2)	32	R/W	Undefined	37.9.4/1536

37.9.1 eTPU Watchdog Timer Register for Engine 1 (eTPU_ER_WDTR_ENG1)

This register configures the watchdog timer for the engine.

Address: 0h base + 60h offset = 60h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R			0													
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R																
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

eTPU_ER_WDTR_ENG1 field descriptions

Field	Description
0–1 WDM	Watchdog Mode. WDM selects the Watchdog operation mode. For more information on the Watchdog operation, see Watchdog. NOTE: Before a new mode is configured, all conditions below must apply: 1. All channels must be disabled (CPR =00). 2. No thread must be executing (eTPU Channel Service Status Register for Engine 1 (eTPU_GCR_CSSR_ENG1) reads 0). 3. The watchdog must be disabled (WDM=00). 00 Disabled 01 Reserved 10 Thread length 11 Busy length
2 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
3 WMO	Watchdog Monitor Operation. This bit enables the monitor operation of the watchdog, inhibiting the thread ending and channel disabling in case of a timeout. The Global Exception occurs on a watchdog timeout, independently of WMO. NOTE: WDSn, WDTO1, WDTO2, and the Global Exception always asserts in case of a watchdog timeout, independently of WMO. 0 Force the end of the running thread and disable the service channel in case of a timeout. 1 Do not force the end of the running thread, nor disable the service channel in case of a timeout.
4–15 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
16–31 WDCNT	Watchdog Count. This field indicates the maximum number of microcycles allowed for a thread (in thread length mode) or a sequence of threads (in busy length mode) before the current running thread is forced to end. For more information on Watchdog operation, see Watchdog. NOTE: The TST microcycles are also counted by the watchdog. NOTE: Fields WDCNT and WMO must not be changed when the watchdog is enabled.

37.9.2 eTPU Idle Register for Engine 1 (eTPU_ER_IDLER_ENG1)

This register counts the microcycles in which the microengine is idle (see [Idle Counter](#)

Address: 0h base + 68h offset = 68h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15															
R	IDLE_CNT																														
W																															
Reset	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*															
Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31															
R	IDLE_CNT																														ICLR
W																															
Reset	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*															

- * Notes:
- x = Undefined at reset.

eTPU_ER_IDLER_ENG1 field descriptions

Field	Description
0–30 IDLE_CNT	Idle Count. This is a freerunning count of the number of idle microcycles in the microengine. For more information on idle counter operation, see Idle Counter . NOTE: When the register is read, the ICLR bit is concatenated with the IDLE_CNT field (as the LSB) to form a 32-bit value.
31 ICLR	Idle Clear. Writing a 1 to this bit clears the IDLE_CNT field. NOTE: When the register is read, the ICLR bit is concatenated with the IDLE_CNT field (as the LSB) to form a 32-bit value. 0 Do not clear the IDLE_CNT field. 1 Clear the IDLE_CNT field.

37.9.3 eTPU Watchdog Timer Register for Engine 2 (eTPU_ER_WDTR_ENG2)

This register configures the watchdog timer for the engine.

Address: 0h base + 70h offset = 70h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R			0								0					
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R																
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

eTPU_ER_WDTR_ENG2 field descriptions

Field	Description
0–1 WDM	Watchdog Mode. WDM selects the Watchdog operation mode. For more information on the Watchdog operation, see Watchdog. NOTE: Before a new mode is configured, all conditions below must apply: 1. All channels must be disabled (CPR =00). 2. No thread must be executing (eTPU Channel Service Status Register for Engine 2 (eTPU_GCR_CSSR_ENG2) reads 0). 3. The watchdog must be disabled (WDM=00). 00 Disabled 01 Reserved 10 Thread length 11 Busy length
2 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
3 WMO	Watchdog Monitor Operation. This bit enables the monitor operation of the watchdog, inhibiting the thread ending and channel disabling in case of a timeout. The Global Exception occurs on a watchdog timeout, independently of WMO. NOTE: WDSn, WDTO1, WDTO2, and the Global Exception always assert in case of a watchdog timeout, independently of WMO. 0 Force the end of the running thread and disable the service channel in case of a timeout. 1 Do not force the end of the running thread, nor disable the service channel in case of a timeout.
4–15 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
16–31 WDCNT	Watchdog Count.

Table continues on the next page...

eTPU_ER_WDTR_ENG2 field descriptions (continued)

Field	Description
	This field indicates the maximum number of microcycles allowed for a thread (in thread length mode) or a sequence of threads (in busy length mode) before the current running thread is forced to end. For more information on Watchdog operation, see Watchdog. NOTE: The TST microcycles are also counted by the watchdog. NOTE: Fields WDCNT and WMO must not be changed when the watchdog is enabled.

37.9.4 eTPU Idle Register for Engine 2 (eTPU_ER_IDLER_ENG2)

This register counts the microcycles in which the microengine is idle (see [Idle Counter](#)).

Address: 0h base + 78h offset = 78h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15															
R	IDLE_CNT																														
W																															
Reset	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*															
Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31															
R	IDLE_CNT																														ICLR
W																															
Reset	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*															

* Notes:

- x = Undefined at reset.

eTPU_ER_IDLER_ENG2 field descriptions

Field	Description
0–30 IDLE_CNT	Idle Count. This is a freerunning count of the number of idle microcycles in the microengine. For more information on idle counter operation, see Idle Counter . NOTE: When the register is read, the ICLR bit is concatenated with the IDLE_CNT field (as the LSB) to form a 32-bit value.
31 ICLR	Idle Clear. Writing a 1 to this bit clears the IDLE_CNT field. NOTE: When the register is read, the ICLR bit is concatenated with the IDLE_CNT field (as the LSB) to form a 32-bit value. 0 Do not clear the IDLE_CNT field. 1 Clear the IDLE_CNT field.

37.10 Latency Monitor memory map and registers

eTPU_LMR memory map

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
80	eTPU Latency Monitor Configuration Register for Engine 1 (eTPU_LMR_LM0CR_ENG1)	32	R/W	0000_0000h	37.10.1/1538
84	eTPU Latency Monitor Measure Register for Engine 1 (eTPU_LMR_LM0MR_ENG1)	32	R/W	0000_0000h	37.10.2/1540
88	eTPU Latency Monitor Configuration Register for Engine 1 (eTPU_LMR_LM1CR_ENG1)	32	R/W	0000_0000h	37.10.1/1538
8C	eTPU Latency Monitor Measure Register for Engine 1 (eTPU_LMR_LM1MR_ENG1)	32	R/W	0000_0000h	37.10.2/1540
90	eTPU Latency Monitor Configuration Register for Engine 1 (eTPU_LMR_LM2CR_ENG1)	32	R/W	0000_0000h	37.10.1/1538
94	eTPU Latency Monitor Measure Register for Engine 1 (eTPU_LMR_LM2MR_ENG1)	32	R/W	0000_0000h	37.10.2/1540
98	eTPU Latency Monitor Configuration Register for Engine 1 (eTPU_LMR_LM3CR_ENG1)	32	R/W	0000_0000h	37.10.1/1538
9C	eTPU Latency Monitor Measure Register for Engine 1 (eTPU_LMR_LM3MR_ENG1)	32	R/W	0000_0000h	37.10.2/1540
A0	eTPU Latency Monitor Configuration Register for Engine 1 (eTPU_LMR_LM4CR_ENG1)	32	R/W	0000_0000h	37.10.1/1538
A4	eTPU Latency Monitor Measure Register for Engine 1 (eTPU_LMR_LM4MR_ENG1)	32	R/W	0000_0000h	37.10.2/1540
A8	eTPU Latency Monitor Configuration Register for Engine 1 (eTPU_LMR_LM5CR_ENG1)	32	R/W	0000_0000h	37.10.1/1538
AC	eTPU Latency Monitor Measure Register for Engine 1 (eTPU_LMR_LM5MR_ENG1)	32	R/W	0000_0000h	37.10.2/1540
C0	eTPU Latency Monitor Configuration Register for Engine 2 (eTPU_LMR_LM0CR_ENG2)	32	R/W	0000_0000h	37.10.3/1541
C4	eTPU Latency Monitor Measure Register for Engine 2 (eTPU_LMR_LM0MR_ENG2)	32	R/W	0000_0000h	37.10.4/1542
C8	eTPU Latency Monitor Configuration Register for Engine 2 (eTPU_LMR_LM1CR_ENG2)	32	R/W	0000_0000h	37.10.3/1541
CC	eTPU Latency Monitor Measure Register for Engine 2 (eTPU_LMR_LM1MR_ENG2)	32	R/W	0000_0000h	37.10.4/1542
D0	eTPU Latency Monitor Configuration Register for Engine 2 (eTPU_LMR_LM2CR_ENG2)	32	R/W	0000_0000h	37.10.3/1541
D4	eTPU Latency Monitor Measure Register for Engine 2 (eTPU_LMR_LM2MR_ENG2)	32	R/W	0000_0000h	37.10.4/1542
D8	eTPU Latency Monitor Configuration Register for Engine 2 (eTPU_LMR_LM3CR_ENG2)	32	R/W	0000_0000h	37.10.3/1541
DC	eTPU Latency Monitor Measure Register for Engine 2 (eTPU_LMR_LM3MR_ENG2)	32	R/W	0000_0000h	37.10.4/1542

Table continues on the next page...

eTPU_LMR memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
E0	eTPU Latency Monitor Configuration Register for Engine 2 (eTPU_LMR_LM4CR_ENG2)	32	R/W	0000_0000h	37.10.3/1541
E4	eTPU Latency Monitor Measure Register for Engine 2 (eTPU_LMR_LM4MR_ENG2)	32	R/W	0000_0000h	37.10.4/1542
E8	eTPU Latency Monitor Configuration Register for Engine 2 (eTPU_LMR_LM5CR_ENG2)	32	R/W	0000_0000h	37.10.3/1541
EC	eTPU Latency Monitor Measure Register for Engine 2 (eTPU_LMR_LM5MR_ENG2)	32	R/W	0000_0000h	37.10.4/1542

37.10.1 eTPU Latency Monitor Configuration Register for Engine 1 (eTPU_LMR_LMnCR_ENG1)

This register configures the operation of individual latency monitors.

Address: 0h base + 80h offset + (8d × i), where i=0d to 5d

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	LMM		0			SRSEL			0			MONCHAN				MAXLAT																
W	LMM					SRSEL						MONCHAN				MAXLAT																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	

eTPU_LMR_LMnCR_ENG1 field descriptions

Field	Description
0–1 LMM	Latency Monitor Mode. This field selects the operation mode of the latency monitor. See the Latency Monitor section for detailed mode descriptions. NOTE: LMM must be in disabled mode to be changed to channel or priority mode. CPR must not be changed while LMM of any Latency Monitor is in priority mode. 00 Disabled mode 01 Channel mode 10 Priority mode 11 Reserved
2–4 Reserved	Reserved. This field is reserved. This read-only field is reserved and always has the value 0.
5–7 SRSEL	Service Request Selection

Table continues on the next page...

eTPU_LMR_LMnCR_ENG1 field descriptions (continued)

Field	Description																							
	This field selects which service requests are monitored. The SRSEL value has different meanings in channel and priority modes, according to the following table. Table 37-43. SRSEL - Service Request Selection <table><tr><th>SRSEL</th><th>Channel Mode</th><th>Priority Mode</th></tr><tr><td>001</td><td>Link SRs monitored only</td><td>Low priority monitored</td></tr><tr><td>010</td><td>Channel SRs monitored only</td><td>Medium priority monitored</td></tr><tr><td>100</td><td>HSRs monitored only</td><td>High priority monitored</td></tr><tr><td>000</td><td>No SRs monitored</td><td rowspan="5">Reserved</td></tr><tr><td>011</td><td>Channel and Link SRs monitored</td></tr><tr><td>101</td><td>HSRs and Link SRs monitored</td></tr><tr><td>110</td><td>HSRs and Channel SRs monitored</td></tr><tr><td>111</td><td>All SRs monitored</td></tr></table>	SRSEL	Channel Mode	Priority Mode	001	Link SRs monitored only	Low priority monitored	010	Channel SRs monitored only	Medium priority monitored	100	HSRs monitored only	High priority monitored	000	No SRs monitored	Reserved	011	Channel and Link SRs monitored	101	HSRs and Link SRs monitored	110	HSRs and Channel SRs monitored	111	All SRs monitored
SRSEL	Channel Mode	Priority Mode																						
001	Link SRs monitored only	Low priority monitored																						
010	Channel SRs monitored only	Medium priority monitored																						
100	HSRs monitored only	High priority monitored																						
000	No SRs monitored	Reserved																						
011	Channel and Link SRs monitored																							
101	HSRs and Link SRs monitored																							
110	HSRs and Channel SRs monitored																							
111	All SRs monitored																							
8–10 Reserved	Reserved. This field is reserved. This read-only field is reserved and always has the value 0.																							
11–15 MONCHAN	Monitored Channel. When the latency monitor is configured for channel mode, this field selects which channel has the latency monitored. MONCHAN has no meaning in priority mode. NOTE: MONCHAN must only be changed when the Latency Monitor is in disabled mode.																							
16–31 MAXLAT	Maximum Latency. This field specifies the maximum latency monitored, in microcycles, before a latency timeout. If MAXLAT=0, the latency keeps on being counted and measured, but the comparison with MAXLAT asserts neither the LTO1/2 fields in eTPU Module Configuration Register (eTPU_SCR_MCR) nor issues a Global Exception. See Latency Monitor for the detailed latency timeout description. NOTE: MAXLAT must only be changed when the Latency Monitor is in disabled mode.																							

37.10.2 eTPU Latency Monitor Measure Register for Engine 1 (eTPU_LMR_LMnMR_ENG1)

This register holds the latency measured value.

Address: 0h base + 84h offset + (8d × i), where i=0d to 5d

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	LATENCY															LTC
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

eTPU_LMR_LMnMR_ENG1 field descriptions

Field	Description
0–15 Reserved	Reserved. This field is reserved. This read-only field is reserved and always has the value 0.
16–30 LATENCY	Latency measurement. This read-only field holds the value of the greatest latency measured, in microcycles, since reset or the latest clear. When the value is greater than the maximum specified in MAXLAT and MAXLAT value is not zero, a Global Exception is issued and LTO1 (engine 1) or LTO2 (engine 2) is set. See Latency Monitor for the detailed latency measurement description. NOTE: When read, the LATENCY field is concatenated with the LTC bit (LSB) to form a 16-bit value. NOTE: Any latency measured above 0xFFFF is captured as 0xFFFF into the LATENCY+LTC field.
31 LTC	Latency measurement clear Writing a 1 to the LTC bit clears the LATENCY field. NOTE: When read, the LATENCY field is concatenated with the LTC bit (LSB) to form a 16-bit value. NOTE: The LATENCY value is not automatically cleared when the Latency Monitor is disabled. 0 Keep the LATENCY field value. 1 Clear the LATENCY field value.

37.10.3 eTPU Latency Monitor Configuration Register for Engine 2 (eTPU_LMR_LMnCR_ENG2)

This register configures the operation of individual latency monitors.

Address: 0h base + C0h offset + (8d × i), where i=0d to 5d

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R																																
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

eTPU_LMR_LMnCR_ENG2 field descriptions

Field	Description																							
0–1 LMM	Latency Monitor Mode. This field selects the operation mode of the latency monitor. See the Latency Monitor section for detailed mode descriptions. NOTE: LMM must be in disabled mode to be changed to channel or priority mode. CPR must not be changed while LMM of any Latency Monitor is in priority mode. 00 Disabled mode 01 Channel mode 10 Priority mode 11 Reserved																							
2–4 Reserved	Reserved. This field is reserved. This read-only field is reserved and always has the value 0.																							
5–7 SRSEL	Service Request Selection This field selects which service requests are monitored. The SRSEL value has different meanings in channel and priority modes, according to the following table. Table 37-64. SRSEL - Service Request Selection <table><tr><th>SRSEL</th><th>Channel Mode</th><th>Channel Mode</th></tr><tr><td>001</td><td>Link SRs monitored only</td><td>Low priority monitored</td></tr><tr><td>010</td><td>Channel SRs monitored only</td><td>Medium priority monitored</td></tr><tr><td>100</td><td>HSRs monitored only</td><td>High priority monitored</td></tr><tr><td>000</td><td>No SRs monitored</td><td rowspan="5">Reserved</td></tr><tr><td>011</td><td>Channel and Link SRs monitored</td></tr><tr><td>101</td><td>HSRs and Link SRs monitored</td></tr><tr><td>110</td><td>HSRs and Channel SRs monitored</td></tr><tr><td>111</td><td>All SRs monitored</td></tr></table>	SRSEL	Channel Mode	Channel Mode	001	Link SRs monitored only	Low priority monitored	010	Channel SRs monitored only	Medium priority monitored	100	HSRs monitored only	High priority monitored	000	No SRs monitored	Reserved	011	Channel and Link SRs monitored	101	HSRs and Link SRs monitored	110	HSRs and Channel SRs monitored	111	All SRs monitored
SRSEL	Channel Mode	Channel Mode																						
001	Link SRs monitored only	Low priority monitored																						
010	Channel SRs monitored only	Medium priority monitored																						
100	HSRs monitored only	High priority monitored																						
000	No SRs monitored	Reserved																						
011	Channel and Link SRs monitored																							
101	HSRs and Link SRs monitored																							
110	HSRs and Channel SRs monitored																							
111	All SRs monitored																							

Table continues on the next page...

eTPU_LMR_LMnCR_ENG2 field descriptions (continued)

Field	Description
8–10 Reserved	Reserved. This field is reserved. This read-only field is reserved and always has the value 0.
11–15 MONCHAN	Monitored Channel. When the latency monitor is configured for channel mode, this field selects which channel has the latency monitored. MONCHAN has no meaning in priority mode. NOTE: MONCHAN must only be changed when the Latency Monitor is in disabled mode.
16–31 MAXLAT	Maximum Latency. This field specifies the maximum latency monitored, in microcycles, before a latency timeout. If MAXLAT=0, the latency keeps on being counted and measured, but the comparison with MAXLAT asserts neither the LTO1/2 fields in eTPU Module Configuration Register (eTPU_SCR_MCR) nor issues a Global Exception. See Latency Monitor for the detailed latency timeout description. NOTE: MAXLAT must only be changed when the Latency Monitor is in disabled mode.

37.10.4 eTPU Latency Monitor Measure Register for Engine 2 (eTPU_LMR_LMnMR_ENG2)

This register holds the latency measured value.

Address: 0h base + C4h offset + (8d × i), where i=0d to 5d

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15															
R	0																														
W																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0															
Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31															
R	LATENCY																														LTC
W																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0															

eTPU_LMR_LMnMR_ENG2 field descriptions

Field	Description
0–15 Reserved	Reserved. This field is reserved. This read-only field is reserved and always has the value 0.
16–30 LATENCY	Latency measurement.

Table continues on the next page...

eTPU_LMR_LMnMR_ENG2 field descriptions (continued)

Field	Description
	This read-only field holds the value of the greatest latency measured, in microcycles, since reset or the latest clear. When the value is greater than the maximum specified in MAXLAT and MAXLAT value is not zero, a Global Exception is issued and LTO1 (engine 1) or LTO2 (engine 2) is set. See Latency Monitor for the detailed latency measurement description. NOTE: When read, the LATENCY field is concatenated with the LTC bit (LSB) to form a 16-bit value. NOTE: Any latency measured above 0xFFFF is captured as 0xFFFF into the LATENCY+LTC field.
31 LTC	Latency measurement clear. Writing a 1 to the LTC bit clears the LATENCY field. NOTE: When read, the LATENCY field is concatenated with the LTC bit (LSB) to form a 16-bit value. NOTE: The LATENCY value is not automatically cleared when the Latency Monitor is disabled. 0 Keep the LATENCY field value. 1 Clear the LATENCY field value.

37.11 Memory Error Support memory map and registers**eTPU_MESR memory map**

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
100	eTPU Memory Error Control Register (eTPU_MESR_MECR)	32	R/W	0000_0000h	37.11.1/1544
104	eTPU Data Error Injection Address Register (eTPU_MESR_DEIAR)	32	R/W	0000_0000h	37.11.2/1547
108	eTPU Data Error Injection Data Pattern Register (eTPU_MESR_DEIDPR)	32	R/W	0000_0000h	37.11.3/1548
10C	eTPU Data Error Injection Parity Pattern Register (eTPU_MESR_DEIPPR)	32	R/W	0000_0000h	37.11.4/1548
110	eTPU Data Error Report Address Register (eTPU_MESR_DERAR)	32	R	0000_0000h	37.11.5/1550
114	eTPU Data Error Report Data Register (eTPU_MESR_DERDR)	32	R	0000_0000h	37.11.6/1551
118	eTPU Data Error Report Syndrome Register (eTPU_MESR_DERSR)	32	R	0000_0000h	37.11.7/1552
120	eTPU Memory Error Status Register (eTPU_MESR_MESR)	32	R/W	0000_0000h	37.11.8/1554
124	eTPU Code Error Injection Address Register (eTPU_MESR_CEIAR)	32	R/W	0000_0000h	37.11.9/1557
128	eTPU Code Error Injection Data Pattern Register (eTPU_MESR_CEIDPR)	32	R/W	0000_0000h	37.11.10/1558
12C	eTPU Code Error Injection Parity Pattern Register (eTPU_MESR_CEIPPR)	32	R/W	0000_0000h	37.11.11/1558

Table continues on the next page...

eTPU_MESR memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
130	eTPU Code Error Report Address Register (eTPU_MESR_CERAR)	32	R	0000_0000h	37.11.12/1559
134	eTPU Code Error Report Data Register (eTPU_MESR_CERDR)	32	R	0000_0000h	37.11.13/1560
138	eTPU Code Error Report Syndrome Register (eTPU_MESR_CERSR)	32	R	0000_0000h	37.11.14/1561
13C	eTPU Code Error Fix Register (eTPU_MESR_CEFR)	32	R/W	0000_0000h	37.11.15/1562

37.11.1 eTPU Memory Error Control Register (eTPU_MESR_MECR)

This register gathers control bits of both SCM and SDM error detection and correction support.

Address: 0h base + 100h offset = 100h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0															MECIE
W	FME															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	
R	HDEIE	MDEIE	0				DTEND	DRDIS	DEDD	CEIE	0				CTEND	CRDIS	CEDD
W										CEIE							
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	

eTPU_MESR_MECR field descriptions

Field	Description
0 FME	Field Modification Enable. Negation of this bit disables the modification of all fields in the register, except MECIE, DRDIS, CRDIS and FME itself: Write accesses do not cause a bus error, regardless of FME state. 0 Fields cannot be modified during a write access, except MECIE, DRDIS, CRDIS and FME. 1 All fields can be modified during a write access.
1–14 Reserved	Reserved.

Table continues on the next page...

eTPU_MESR_MECR field descriptions (continued)

Field	Description
	This field is reserved. This read-only field is reserved and always has the value 0.
15 MECIE	Memory Error Correction Interrupt Enable. This bit enables the Memory Error Correction Interrupt (see Interrupt types and sources). 0 Memory Error Correction Interrupt disabled. 1 Memory Error Correction Interrupt enabled.
16 HDEIE	Host Data Error Injection Enable This bit enables error injection into the SDM host read path, including CDC transfers. The eTPU Data Error Injection Address Register (eTPU_MESR_DEIAR) specifies the address where the error is injected, while eTPU Data Error Injection Data Pattern Register (eTPU_MESR_DEIDPR) and eTPU Data Error Injection Parity Pattern Register (eTPU_MESR_DEIPPR) define the bit flips on data and parity, respectively. NOTE: Data error injection is also performed when data error detection is disabled (DEDD=1). 0 SDM error injection disabled for host reads. 1 SDM error injection enabled for host reads.
17 MDEIE	Microengine(s) Data Error Injection Enable. This bit enables error injection into the SDM microengine read path. The eTPU Data Error Injection Address Register (eTPU_MESR_DEIAR) specifies the address where the error is injected, while eTPU Data Error Injection Data Pattern Register (eTPU_MESR_DEIDPR) and eTPU Data Error Injection Parity Pattern Register (eTPU_MESR_DEIPPR) define the bit flips on data and parity, respectively. NOTE: Data error injection is also performed when data error detection is disabled (DEDD=1). 0 SDM error injection disabled for microengine reads. 1 SDM error injection enabled for microengine reads.
18–20 Reserved	Reserved. This field is reserved. This read-only field is reserved and always has the value 0.
21 DTEND	Thread End on Data Non-correctable Error. This bit controls the behavior of the microengine when a non-correctable data error occurs. For more details, see Memory error support. NOTE: DTEND must only be changed when the engine is stopped or halted, or when all channels are disabled. 0 Thread continues on non-correctable data read access by the microengine. 1 Thread ends on non-correctable data read access by the microengine.
22 DRDIS	Data Error Report Disable. This bit disables the sampling of SDM error address, data and syndrome into the eTPU Memory Error Support Registers, and eTPU Data Error Report Syndrome Register (eTPU_MESR_DERSR). It does not, however, disable the setting of eTPU Memory Error Status Register (eTPU_MESR_MESR) bits DCERR and DNCERR.

Table continues on the next page...

eTPU_MESR_MECR field descriptions (continued)

Field	Description
	NOTE: To guarantee coherent reading of eTPU Data Error Report Address Register (eTPU_MESR_DERAR), eTPU Data Error Report Data Register (eTPU_MESR_DERDR), and eTPU Data Error Report Syndrome Register (eTPU_MESR_DERSR), DRDIS must be asserted beforehand if SDM concurrent accesses can happen, by host or microengine. After the registers are read, DRDIS may be negated again to allow new samplings. 0 SDM error report sampling enabled. 1 SDM error report sampling disabled.
23 DEDD	Data Error Detection Disable. This bit disables SDM error detection. The disabled state turns ineffective any action due to error detection, including consequently error correction. 0 SDM error detection enabled. 1 SDM error detection disabled.
24 CEIE	Code Error Injection Enable. This bit enables error injection into the SCM read path, either on microengine accesses (see VIS =0) or host accesses (VIS = 1) eTPU Code Error Injection Address Register (eTPU_MESR_CEIAR) specifies the address where the error is injected, while eTPU Code Error Injection Data Pattern Register (eTPU_MESR_CEIDPR) and eTPU Code Error Injection Parity Pattern Register (eTPU_MESR_CEIPPR) define the bit flips on data and parity, respectively. NOTE: Code error injection is also performed when code error detection is disabled (CEDD=1). 0 SCM error injection disabled. 1 SCM error injection enabled.
25–28 Reserved	Reserved. This field is reserved. This read-only field is reserved and always has the value 0.
29 CTEND	Thread End on Code Non-correctable Error. This bit controls the behavior of the microengine when a non-correctable code error occurs. For more details, see Memory error support. NOTE: CTEND must only be changed when the engine is stopped or halted, or when all channels are disabled. 0 Thread continues on instruction fetch non-correctable error. 1 Thread ends on instruction fetch non-correctable error.
30 CRDIS	Code Error Report Disable. This bit disables the sampling of SCM error address, data and syndrome into the eTPU Code Error Report Address Register (eTPU_MESR_CERAR), eTPU Code Error Report Data Register (eTPU_MESR_CERDR), and eTPU Code Error Report Syndrome Register (eTPU_MESR_CERSR). It does not, however, disable the setting of CCERR and CNCERR in eTPU Memory Error Status Register (eTPU_MESR_MESR).

Table continues on the next page...

eTPU_MESR_MECR field descriptions (continued)

Field	Description
	NOTE: To guarantee coherent reading of eTPU Code Error Report Address Register (eTPU_MESR_CERAR), eTPU Code Error Report Data Register (eTPU_MESR_CERDR), and eTPU Code Error Report Syndrome Register (eTPU_MESR_CERSR), CRDIS must be asserted beforehand if SCM concurrent accesses can happen, by host or microengine. After the registers are read, CRDIS may be negated again to allow new samplings. 0 SCM error report sampling enabled. 1 SCM error report sampling disabled.
31 CEDD	Code Error Detection Disable. This bit disables SCM error detection. The disabled state turns ineffective any action due to error detection, including consequently error correction. 0 SCM error detection enabled. 1 SCM error detection disabled.

37.11.2 eTPU Data Error Injection Address Register (eTPU_MESR_DEIAR)

This register holds the address where error is to be injected into the SDM read data path. The two least significant bits are 0 (read-only), so that the address is always word-aligned.

NOTE

This register must only be modified when [HDEIE](#) and [MDEIE](#) are both negated.

Address: 0h base + 104h offset = 104h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0																INJ_ADDR														0	
W																		INJ_ADDR														
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	

eTPU_MESR_DEIAR field descriptions

Field	Description
0–17 Reserved	Reserved. This field is reserved. This read-only field is reserved and always has the value 0.
18–29 INJ_ADDR	Injection Address. INJ_ADDR holds the (word) address where an error is injected into the SDM read data path, as specified by eTPU Data Error Injection Data Pattern Register (eTPU_MESR_DEIDPR) and eTPU Data Error

Table continues on the next page...

eTPU_MESR_DEIAR field descriptions (continued)

Field	Description
	Injection Parity Pattern Register (eTPU_MESR_DEIPPR) . For more information on error injection, see Error injection .
30–31 Reserved	Reserved. This field is reserved. This read-only field is reserved and always has the value 0.

37.11.3 eTPU Data Error Injection Data Pattern Register (eTPU_MESR_DEIDPR)

This register is used to specify the bits in which error is injected into the SDM read data path.

NOTE

This register must only be modified when [HDEIE](#) and [MDEIE](#) are both negated.

Address: 0h base + 108h offset = 108h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R																																
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

eTPU_MESR_DEIDPR field descriptions

Field	Description
0–31 DFLIP	Data Flip bits. Each bit corresponds to a bit in the actual data read from the SDM address defined in eTPU Data Error Injection Address Register (eTPU_MESR_DEIAR), specifying whether it should be inverted or not. The inversion only occurs if the SDM error injection is enabled by HDEIE and/or MDEIE in eTPU Memory Error Control Register (eTPU_MESR_MECR). For more information on error injection, see Error injection. 0 Read data bit is not inverted. 1 Read data bit is inverted.

37.11.4 eTPU Data Error Injection Parity Pattern Register (eTPU_MESR_DEIPPR)

This register is used to specify the bits in which error is injected into the SDM read parity path.

NOTE

This register must only be modified when [HDEIE](#) and [MDEIE](#) are both negated.

Address: 0h base + 10Ch offset = 10Ch

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0								0								0								0							
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

eTPU_MESR_DEIPPR field descriptions

Field	Description
0–2 Reserved	Reserved. This field is reserved. This read-only field is reserved and always has the value 0.
3–7 PFLIP_19_15	Parity Flip bits. Each bit corresponds to a bit in the actual parity read from the SDM address defined in eTPU Data Error Injection Address Register (eTPU_MESR_DEIAR) , specifying whether it should be inverted or not. The inversion only occurs if the SDM error injection is enabled by the HDEIE and/or MDEIE fields in eTPU Memory Error Control Register (eTPU_MESR_MECCR) . Bits PFLIP[19:15] corresponds to the parity bits for the most significant byte, PFLIP[14:10] to the next most significant byte and so on until PFLIP[4:0] for the least significant byte. For more information on error injection, see Error injection . 0 Parity data bit is not inverted. 1 Parity data bit is inverted.
8–10 Reserved	Reserved. This field is reserved. This read-only field is reserved and always has the value 0.
11–15 PFLIP_14_10	Parity Flip bits. Each bit corresponds to a bit in the actual parity read from the SDM address defined in eTPU Data Error Injection Address Register (eTPU_MESR_DEIAR) , specifying whether it should be inverted or not. The inversion only occurs if the SDM error injection is enabled by the HDEIE and/or MDEIE fields in eTPU Memory Error Control Register (eTPU_MESR_MECCR) . Bits PFLIP[19:15] corresponds to the parity bits for the most significant byte, PFLIP[14:10] to the next most significant byte and so on until PFLIP[4:0] for the least significant byte. For more information on error injection, see Error injection . 0 Parity data bit is not inverted. 1 Parity data bit is inverted.
16–18 Reserved	Reserved. This field is reserved. This read-only field is reserved and always has the value 0.
19–23 PFLIP_9_5	Parity Flip bits. Each bit corresponds to a bit in the actual parity read from the SDM address defined in eTPU Data Error Injection Address Register (eTPU_MESR_DEIAR) , specifying whether it should be inverted or not. The inversion only occurs if the SDM error injection is enabled by the HDEIE and/or MDEIE fields in eTPU Memory Error Control Register (eTPU_MESR_MECCR) . Bits PFLIP[19:15] corresponds to the parity bits for

Table continues on the next page...

eTPU_MESR_DEIPPR field descriptions (continued)

Field	Description
	the most significant byte, PFLIP[14:10] to the next most significant byte and so on until PFLIP[4:0] for the least significant byte. For more information on error injection, see Error injection . 0 Parity data bit is not inverted. 1 Parity data bit is inverted.
24–26 Reserved	Reserved. This field is reserved. This read-only field is reserved and always has the value 0.
27–31 PFLIP_4_0	Parity Flip bits. Each bit corresponds to a bit in the actual parity read from the SDM address defined in eTPU Data Error Injection Address Register (eTPU_MESR_DEIAR) , specifying whether it should be inverted or not. The inversion only occurs if the SDM error injection is enabled by the HDEIE and/or MDEIE fields in eTPU Memory Error Control Register (eTPU_MESR_MECCR) . Bits PFLIP[19:15] corresponds to the parity bits for the most significant byte, PFLIP[14:10] to the next most significant byte and so on until PFLIP[4:0] for the least significant byte. For more information on error injection, see Error injection . 0 Parity data bit is not inverted. 1 Parity data bit is inverted.

37.11.5 eTPU Data Error Report Address Register (eTPU_MESR_DERAR)

This register holds the address where the last non-correctable error was detected from the SDM. It also captures the address of the last correctable error detected, as long as [DNCERR](#) is 0. The two least significant bits are 0 (read-only), so that the address is always word-aligned.

Address: 0h base + 110h offset = 110h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	ERR_ACC	0	ERR_CHANNEL						0							
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	ERR_ADDR														0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

eTPU_MESR_DERAR field descriptions

Field	Description
0–1 ERR_ACC	Error Access Source Indication. This two-bit field indicates the source of the access when the error occurred. 00 Host - direct access 01 Engine 1 10 Engine 2 11 CDC
2 Reserved	Reserved. This field is reserved. This read-only field is reserved and always has the value 0.
3–7 ERR_CHANNEL	Error Channel Access Indication. This field holds the number of the channel being serviced (not necessarily CHAN) when the error occurred. Its content is meaningful only when the field ERR_ACC indicates Engine 1 or Engine 2.
8–17 Reserved	Reserved. This field is reserved. This read-only field is reserved and always has the value 0.
18–29 ERR_ADDR	Error Address. ERR_ADDR holds the (word) address where the last captured SDM read error was detected. The last captured error is a non-correctable one if DNCERR =1, or a correctable one if DNCERR =0 and DCERR =1. The field is meaningless if both DNCERR and DCERR are 0. For more information on error detection, see Memory error support .
30–31 Reserved	Reserved. This field is reserved. This read-only field is reserved and always has the value 0.

37.11.6 eTPU Data Error Report Data Register (eTPU_MESR_DERDR)

This register holds the data read with the last non-correctable error detected from the SDM. It also captures the data of the last correctable error read detection, as long as **DNCERR** is 0. In both cases, the data is captured after the error correction logic. The data is meaningless if both **DNCERR** and **DCERR** are 0. For more information on error detection, see [Memory error support](#).

NOTE

Data is sampled before the error correction logic.

Address: 0h base + 114h offset = 114h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	DATA																															
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

eTPU_MESR_DERDR field descriptions

Field	Description
0–31 DATA	DATA. NOTE: Individual data bytes are not meaningful when their respective BE bits in eTPU Data Error Report Syndrome Register (eTPU_MESR_DERSR) are not asserted.

37.11.7 eTPU Data Error Report Syndrome Register (eTPU_MESR_DERSR)

This register holds the syndrome read with the last non-correctable error detected from the SDM. It also captures the syndrome of the last correctable error read detection, as long as [DNCERR](#) is 0.

Address: 0h base + 118h offset = 118h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	BE3	0							BE2	0						
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	BE1	0							BE0	0						
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

eTPU_MESR_DERSR field descriptions

Field	Description
0 BE3	Byte Enable 3. Each BE bit corresponds to a report byte in a 32-bit aligned data word. BE3 corresponds to the most significant one. BE qualifies the error report for its respective byte. 0 Byte was not selected for read, therefore its respective error report fields are not meaningful. 1 Byte was selected for read, and the error report for the respective byte is meaningful.
1–2 Reserved	Reserved. This field is reserved. This read-only field is reserved and always has the value 0.
3–7 SYND3	Error Syndrome 3. SYND3 holds the syndrome of an error detected in the SDM read data path, as defined in Error Correction Code (ECC) and syndrome definition . The field is meaningless if both DNCERR and DCERR are 0. For more information on error detection, see Memory error support .

Table continues on the next page...

eTPU_MESR_DERSR field descriptions (continued)

Field	Description
	NOTE: Syndrome fields which have their respective BE bits negated are not meaningful.
8 BE2	Byte Enable 2. Each BE bit corresponds to a report byte in a 32-bit aligned data word. BE3 corresponds to the most significant one. BE qualifies the error report for its respective byte. 0 Byte was not selected for read, therefore its respective error report fields are not meaningful. 1 Byte was selected for read, and the error report for the respective byte is meaningful.
9–10 Reserved	Reserved. This field is reserved. This read-only field is reserved and always has the value 0.
11–15 SYND2	Error Syndrome 2. SYND2 holds the syndrome of an error detected in the SDM read data path, as defined in the “Error Correction Code (ECC) and Syndrome Definition” section. The field is meaningless if both DNCERR and DCERR are 0. For more information on error detection, see Memory error support. NOTE: Syndrome fields which have their respective BE bits negated are not meaningful.
16 BE1	Byte Enable 1. Each BE bit corresponds to a report byte in a 32-bit aligned data word. BE3 corresponds to the most significant one. BE qualifies the error report for its respective byte. 0 Byte was not selected for read, therefore its respective error report fields are not meaningful. 1 Byte was selected for read, and the error report for the respective byte is meaningful.
17–18 Reserved	Reserved. This field is reserved. This read-only field is reserved and always has the value 0.
19–23 SYND1	Error Syndrome 1. SYND1 holds the syndrome of an error detected in the SDM read data path, as defined in the “Error Correction Code (ECC) and Syndrome Definition” section. The field is meaningless if both DNCERR and DCERR are 0. For more information on error detection, see Memory error support. NOTE: Syndrome fields which have their respective BE bits negated are not meaningful.
24 BE0	Byte Enable 0. Each BE bit corresponds to a report byte in a 32-bit aligned data word. BE3 corresponds to the most significant one. BE qualifies the error report for its respective byte. 0 Byte was not selected for read, therefore its respective error report fields are not meaningful. 1 Byte was selected for read, and the error report for the respective byte is meaningful.
25–26 Reserved	Reserved. This field is reserved. This read-only field is reserved and always has the value 0.
27–31 SYND0	Error Syndrome 0.

Table continues on the next page...

eTPU_MESR_DERSR field descriptions (continued)

Field	Description
	SYND0 holds the syndrome of an error detected in the SDM read data path, as defined in the “Error Correction Code (ECC) and Syndrome Definition” section. The field is meaningless if both DNCERR and DCERR are 0. For more information on error detection, see Memory error support .
	NOTE: Syndrome fields which have their respective BE bits negated are not meaningful.

37.11.8 eTPU Memory Error Status Register (eTPU_MESR_MESR)

This register gathers status bits of both SCM and SDM error detection and correction support.

Address: 0h base + 120h offset = 120h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R																
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R																
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

eTPU_MESR_MESR field descriptions

Field	Description
0 DCERR	Data Correctable Error flag. When read: 0.....No SDM correctable error occurred. 1.....An SDM correctable error occurred. Writing 1 to this bit clears the DCERR status flag. DCERR asserts even if DNCERR is asserted beforehand, but in this case the address and data of the erroneous access is not captured (see Memory Error Support memory map and registers and following sections). DCERR is cleared by writing a 1 to it. Assertion of this bit can also assert the Memory Error Correction Interrupt (see Interrupt types and sources). NOTE: A correctable error is not considered for flagging if accompanied of a non-correctable one. Therefore, if both correctable and non-correctable data errors occur on the same access (in distinct bytes), the DNCERR flag asserts, but not the DCERR flag.

Table continues on the next page...

eTPU_MESR_MESR field descriptions (continued)

Field	Description
	0 Do not clear DCERR flag. 1 Clears DCERR flag
1–6 Reserved	Reserved. This field is reserved. This read-only field is reserved and always has the value 0.
7 DCOVR	Data Correctable Overrun flag. Writing 1 to this bit clears the DCOVR status flag. 0 Data correctable error occurred while DCOVR = 1. 1 No data correctable error occurred while DCOVR = 1. DCOVR asserts when an SDM correctable error occurs and the DCERR flag is already asserted. Assertion of this bit causes no interrupt. NOTE: A correctable error is not considered for flagging if accompanied by a non-correctable one. Therefore, if both correctable and non-correctable data errors occur on the same access (in distinct bytes), the DCOVR does not assert, even if DCERR is asserted. The DNCOVR flag, on the other hand, asserts if DNCERR is asserted.
8 CCERR	Code Error flag. When read: 0.....No SCM correctable error occurred. 1.....An SCM correctable error occurred. When written: 0.....Do not clear CCERR flag. 1.....Clears CCERR flag. The CCERR flag asserts even if CNCERR is asserted beforehand, but in this case the address and data of the erroneous access is not captured (see eTPU Code Error Report Address Register (eTPU_MESR_CERAR) and the following sections). CCERR is cleared by writing a 1 at the same position. Assertion of this bit can also assert the Memory Error Correction Interrupt (see Interrupt types and sources).
9–14 Reserved	Reserved. This field is reserved. This read-only field is reserved and always has the value 0.
15 CCOVR	Code Correctable Overrun flag This bit asserts when an SCM correctable error occurs and the CCERR flag is already asserted. Assertion of this bit causes no interrupt. When read: 0.....No SCM correctable error overrun occurred. 1.....An SCM correctable error overrun occurred. Writing 1 to this bit clears the CCOVR status flag: 0 Do not clear CCOVR flag. 1 Clear CCOVR flag.

Table continues on the next page...

eTPU_MESR_MESR field descriptions (continued)

Field	Description
16 DNCERR	Data Non-Correctable Error flag DNCERR flags a non-correctable error. When read: 0.....No SDM non-correctable error occurred. 1.....An SDM non-correctable error occurred; registers eTPU Global Channel Registers are not updated by correctable error detections. DNCERR must be cleared (by writing a 1 to it) to allow correctable error updates to eTPU Data Error Report Address Register (eTPU_MESR_DERAR) and eTPU Data Error Report Data Register (eTPU_MESR_DERDR). 0 Do not clear DNCERR flag. 1 Clear DNCERR flag.
17–22 Reserved	Reserved. This field is reserved. This read-only field is reserved and always has the value 0.
23 DNCOVR	Data Non-Correctable Overrun flag. DNCOVR asserts when an SDM non-correctable error occurs and the DNCERR flag is already asserted. Assertion of this bit causes no interrupt. When read: 0.....No SDM non-correctable error overrun occurred. 1.....An SDM non-correctable error overrun occurred. DNCOVR is cleared by writing a 1 to it. 0 Do not clear DNCOVR flag. 1 Clear DNCOVR flag.
24 CNCERR	Code Non-Correctable Error flag CNCERR flags a non-correctable error: When read: 0.....An SCM non-correctable error occurred; the eTPU Global Channel Registers are not updated by correctable error detections. 1.....an SCM non-correctable error occurred; eTPU Code Error Report Address Register (eTPU_MESR_CERAR) and eTPU Code Error Report Data Register (eTPU_MESR_CERDR) are not updated by correctable error detections. CNCERR must be cleared to allow correctable error updates to eTPU Code Error Report Address Register (eTPU_MESR_CERAR) and eTPU Code Error Report Data Register (eTPU_MESR_CERDR). CNCERR is cleared by writing a 1 to it. 0 Do not clear CNCERR flag. 1 Clear CNCERR flag.
25–30 Reserved	Reserved. This field is reserved. This read-only field is reserved and always has the value 0.

Table continues on the next page...

eTPU_MESR_MESR field descriptions (continued)

Field	Description
31 CNCOVR	Code Non-Correctable Overrun flag. CNCOVR asserts when an SCM non-correctable error occurs and the CNCERR flag is already asserted. Assertion of this bit causes no interrupt. When read: 0.....an SCM non-correctable error overrun occurred. 1.....An SCM non-correctable error overrun occurred. When written: 0 Do not clear CNCOVR flag. 1 Clear CNCOVR flag.

37.11.9 eTPU Code Error Injection Address Register (eTPU_MESR_CEIAR)

This register holds the address where error is to be injected into the SCM read data path. The two least significant bits are 0 (read-only), so that the address is always word-aligned.

NOTE

This register must only be modified when [CEIE](#) is negated.

Address: 0h base + 124h offset = 124h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0																INJ_ADDR														0	
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	

eTPU_MESR_CEIAR field descriptions

Field	Description
0–15 Reserved	Reserved. This field is reserved. This read-only field is reserved and always has the value 0.
16–29 INJ_ADDR	Injection Address. INJ_ADDR holds the (word) address where an error is injected in the SCM read data path, as specified by eTPU Code Error Injection Data Pattern Register (eTPU_MESR_CEI DPR) and eTPU Code Error Injection Parity Pattern Register (eTPU_MESR_CEI PPR). For more information on error injection, see Error injection.
30–31 Reserved	Reserved. This field is reserved. This read-only field is reserved and always has the value 0.

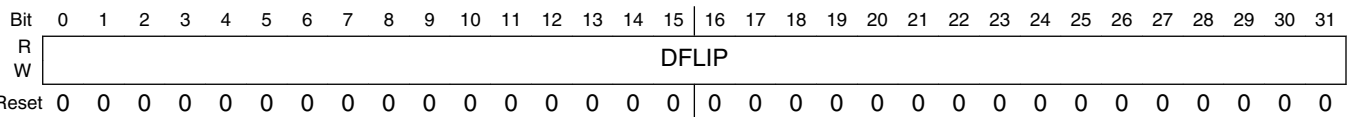
37.11.10 eTPU Code Error Injection Data Pattern Register (eTPU_MESR_CEIDPR)

This register is used to specify the bits in which error is injected into the SCM read data path.

NOTE

This register must only be modified when CEIE is negated.

Address: 0h base + 128h offset = 128h



eTPU_MESR_CEIDPR field descriptions

Field	Description
0–31 DFLIP	Data Flip bits. Each bit corresponds to a bit in the actual data read from the SCM address defined in eTPU Code Error Injection Address Register (eTPU_MESR_CEIAR), specifying whether it should be inverted or not. The inversion only occurs if the SCM error injection is enabled by CEIE. For more information on error injection, see Error injection. 0 Read data bit is not inverted. 1 Read data bit is inverted.

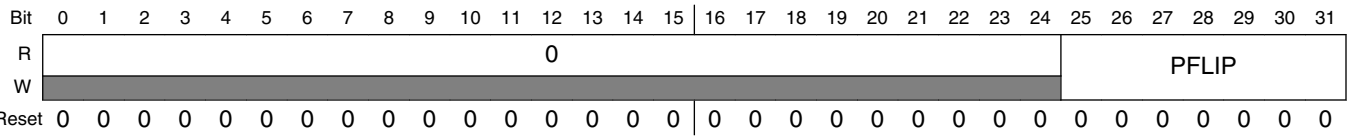
37.11.11 eTPU Code Error Injection Parity Pattern Register (eTPU_MESR_CEIPPR)

This register is used to specify the bits in which error is injected into the SCM read parity path.

NOTE

This register must only be modified when CEIE is negated.

Address: 0h base + 12Ch offset = 12Ch



eTPU_MESR_CEIPPR field descriptions

Field	Description
0–24 Reserved	Reserved. This field is reserved. This read-only field is reserved and always has the value 0.
25–31 PFLIP	Parity Flip bits. Each bit corresponds to a bit in the actual parity/code read from the SCM address defined in eTPU Code Error Injection Address Register (eTPU_MESR_CEIAR) , specifying whether it should be inverted or not. The inversion only occurs if the SCM error injection is enabled by CEIE . For more information on error injection, see Error injection . 0 Parity data bit is not inverted. 1 Parity data bit is inverted.

37.11.12 eTPU Code Error Report Address Register (eTPU_MESR_CERAR)

This register holds the address where the last non-correctable error was detected from the SCM. It also captures the address of the last correctable error detected, as long as [CNCERR](#) is 0. The two least significant bits are 0 (read-only), so that the address is always word-aligned.

Address: 0h base + 130h offset = 130h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	ERR_ACC		0	ERR_CHANNEL				0								
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	ERR_ADDR														0	
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

eTPU_MESR_CERAR field descriptions

Field	Description
0–1 ERR_ACC	Error Access Source Indication. This two-bit field indicates the source of the access when the error occurred. 00 Host 01 Engine 1

Table continues on the next page...

eTPU_MESR_CERAR field descriptions (continued)

Field	Description
	10 Engine 2 11 MISC
2 Reserved	Reserved. This field is reserved. This read-only field is reserved and always has the value 0.
3–7 ERR_CHANNEL	Error Channel Access Indication. This field holds the number of the channel being serviced (not necessarily CHAN) when the error occurred. Its content is meaningful only when the field ERR_ACC indicates Engine 1 or Engine 2.
8–15 Reserved	Reserved. This field is reserved. This read-only field is reserved and always has the value 0.
16–29 ERR_ADDR	Error Address. ERR_ADDR holds the (word) address where the last captured SCM read error was detected. The last captured error is a non-correctable one if CNCERR = 1, or a correctable one if CNCERR = 0 and CCERR = 1. The field is meaningless if both CNCERR and CCERR are 0. For more information on error detection, see Memory error support .
30–31 Reserved	Reserved. This field is reserved. This read-only field is reserved and always has the value 0.

37.11.13 eTPU Code Error Report Data Register (eTPU_MESR_CERDR)

This register holds the data read with the last non-correctable error detected from the SCM. It also captures the data of the last correctable error read detection, as long as **CNCERR** is 0. In both cases, the data is captured after the error correction logic. The data may be meaningless if both **CNCERR** and **CCERR** are 0. For more information on error detection, see [Memory error support](#).

NOTE

CODE is sampled before the error correction logic.

Address: 0h base + 134h offset = 134h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	CODE																															
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

eTPU_MESR_CERDR field descriptions

Field	Description
0–31 CODE	CODE[31-0].

37.11.14 eTPU Code Error Report Syndrome Register (eTPU_MESR_CERSR)

This register holds the syndrome and global parity read with the last non-correctable error detected from the SCM. It also captures the syndrome of the last correctable error read detection, as long as [CNCERR](#) is 0.

Address: 0h base + 138h offset = 138h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0																SYND															
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	

eTPU_MESR_CERSR field descriptions

Field	Description
0–24 Reserved	Reserved. This field is reserved. This read-only field is reserved and always has the value 0.
25–31 SYND	Error Syndrome. SYND holds the syndrome of an error detected in the SCM read data path, as defined in Error Correction Code (ECC) and syndrome definition . The field is meaningless if both CNCERR and CCERR are 0. For more information on error detection, see Memory error support .

37.11.15 eTPU Code Error Fix Register (eTPU_MESR_CEFR)

This register is used to fix single-bit error in a specified SCM position. The fix is done automatically, concurrently with code execution, by the MISC logic (see [Error fixing](#)).

Address: 0h base + 13Ch offset = 13Ch

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	CFIXM		0													
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	CFIX_ADDR														0	
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

eTPU_MESR_CEFR field descriptions

Field	Description															
0–1 CFIXM	Code Fix Mode. This field specifies the SCM MISC fix logic operation mode as shown in the table below. NOTE: The CFIXM field can only be changed to/from disabled mode. Table 37-100. CFIXM - Code Fix Mode <table><tr><th>Value</th><th>Mode</th><th>Description</th></tr><tr><td>00</td><td>Disabled</td><td>SCM error fix disabled.</td></tr><tr><td>01</td><td>Reserved</td><td>Reserved value, must not be used.</td></tr><tr><td>10</td><td>Reserved</td><td>Reserved value, must not be used.</td></tr><tr><td>11</td><td>Automatic</td><td>The MISC logic rewrites the code word value read (already fixed by the error correction mechanism) into the same address whenever it finds a correctable error.</td></tr></table>	Value	Mode	Description	00	Disabled	SCM error fix disabled.	01	Reserved	Reserved value, must not be used.	10	Reserved	Reserved value, must not be used.	11	Automatic	The MISC logic rewrites the code word value read (already fixed by the error correction mechanism) into the same address whenever it finds a correctable error.
Value	Mode	Description														
00	Disabled	SCM error fix disabled.														
01	Reserved	Reserved value, must not be used.														
10	Reserved	Reserved value, must not be used.														
11	Automatic	The MISC logic rewrites the code word value read (already fixed by the error correction mechanism) into the same address whenever it finds a correctable error.														
2–15 Reserved	Reserved. This field is reserved. This read-only field is reserved and always has the value 0.															
16–29 CFIX_ADDR	Code Fix Address. This field is used for hardware debug purposes. CFIX_ADDR may change its contents and must not be written in automatic mode. For more information on error detection and correction, see Memory error support.															
30–31 Reserved	Reserved. This field is reserved. This read-only field is reserved and always has the value 0.															

37.12 Global Channel memory map and registers

The registers in this section group, by type, the interrupt status and enable bits from all the channels. This organization eases management of all channels or groups of channels by a single interrupt handler routine. These bits, except the service and watchdog status, are mirrored in the individual channel registers, grouped by channel.

eTPU_GCR memory map

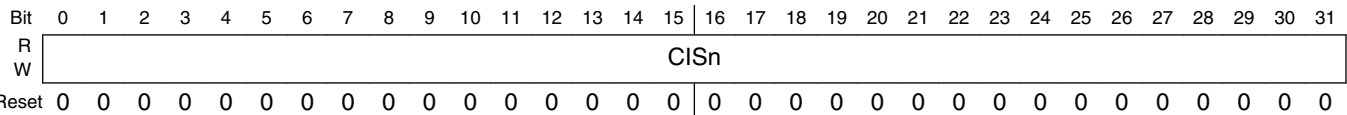
Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
200	eTPU Channel Interrupt Status Register for Engine 1 (eTPU_GCR_CISR_ENG1)	32	R/W	0000_0000h	37.12.1/1564
204	eTPU Channel Interrupt Status Register for Engine 2 (eTPU_GCR_CISR_ENG2)	32	R/W	0000_0000h	37.12.2/1564
210	eTPU Channel Data Transfer Request Status Register for Engine 1 (eTPU_GCR_CDTRSR_ENG1)	32	R/W	0000_0000h	37.12.3/1565
214	eTPU Channel Data Transfer Request Status Register for Engine 2 (eTPU_GCR_CDTRSR_ENG2)	32	R/W	0000_0000h	37.12.4/1566
220	eTPU Channel Interrupt Overflow Status Register for Engine 1 (eTPU_GCR_CIOSR_ENG1)	32	R/W	0000_0000h	37.12.5/1566
224	eTPU Channel Interrupt Overflow Status Register for Engine 2 (eTPU_GCR_CIOSR_ENG2)	32	R/W	0000_0000h	37.12.6/1567
230	eTPU Channel Data Transfer Request Overflow Status Register for Engine 1 (eTPU_GCR_CDTROSR_ENG1)	32	R/W	0000_0000h	37.12.7/1568
234	eTPU Channel Data Transfer Request Overflow Status Register for Engine 2 (eTPU_GCR_CDTROSR_ENG2)	32	R/W	0000_0000h	37.12.8/1568
240	eTPU Channel Interrupt Enable Register for Engine 1 (eTPU_GCR_CIER_ENG1)	32	R/W	0000_0000h	37.12.9/1569
244	eTPU Channel Interrupt Enable Register for Engine 2 (eTPU_GCR_CIER_ENG2)	32	R/W	0000_0000h	37.12.10/1569
250	eTPU Channel Data Transfer Request Enable Register for Engine 1 (eTPU_GCR_CDTRER_ENG1)	32	R/W	0000_0000h	37.12.11/1570
254	eTPU Channel Data Transfer Request Enable Register for Engine 2 (eTPU_GCR_CDTRER_ENG2)	32	R/W	0000_0000h	37.12.12/1570
260	eTPU Watchdog Status Register for Engine 1 (eTPU_GCR_WDSR_ENG1)	32	R/W	0000_0000h	37.12.13/1571
264	eTPU Watchdog Status Register for Engine 2 (eTPU_GCR_WDSR_ENG2)	32	R/W	0000_0000h	37.12.14/1572
280	eTPU Channel Pending Service Status Register for Engine 1 (eTPU_GCR_CPSSR_ENG1)	32	R	0000_0000h	37.12.15/1572
284	eTPU Channel Pending Service Status Register for Engine 2 (eTPU_GCR_CPSSR_ENG2)	32	R	0000_0000h	37.12.16/1573
290	eTPU Channel Service Status Register for Engine 1 (eTPU_GCR_CSSR_ENG1)	32	R	0000_0000h	37.12.17/1574
294	eTPU Channel Service Status Register for Engine 2 (eTPU_GCR_CSSR_ENG2)	32	R	0000_0000h	37.12.18/1574

37.12.1 eTPU Channel Interrupt Status Register for Engine 1 (eTPU_GCR_CISR_ENG1)

This register groups the host interrupt status from all channels. These bits are mirrored from the Channel Status Control register (see [eTPU Channel n Status Control Register for Engine 1 \(eTPU_CCCR_CnSCR_ENG1\)](#)) and the Host must write 1 to clear a status bit.

For details about interrupts, see [Interrupts and data transfer requests](#).

Address: 0h base + 200h offset = 200h



eTPU_GCR_CISR_ENG1 field descriptions

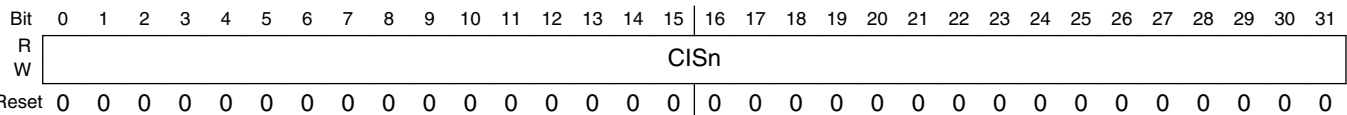
Field	Description
0–31 CISn	Channel n Interrupt Status. When read, each bit: 0.....Indicates that channel n has no pending interrupt to the CPU. 1.....Indicates that channel n has a pending interrupt to the CPU. When written: 0.....Keep interrupt status bit unaltered. 1.....Clear interrupt status bit.

37.12.2 eTPU Channel Interrupt Status Register for Engine 2 (eTPU_GCR_CISR_ENG2)

This register groups the host interrupt status from all channels. These bits are mirrored from the Channel Status Control register (see [eTPU Channel n Status Control Register for Engine 2 \(eTPU_CCCR_CnSCR_ENG2\)](#)) and the Host must write 1 to clear a status bit.

For details about interrupts, see [Interrupts and data transfer requests](#).

Address: 0h base + 204h offset = 204h



eTPU_GCR_CISR_ENG2 field descriptions

Field	Description
0–31 CISn	Channel n Interrupt Status. When read, each bit: 0.....Indicates that channel n has no pending interrupt to the CPU. 1.....Indicates that channel n has a pending interrupt to the CPU. When written: 0.....Keep interrupt status bit unaltered. 1.....Clear interrupt status bit.

37.12.3 eTPU Channel Data Transfer Request Status Register for Engine 1 (eTPU_GCR_CDTRSR_ENG1)

This register groups data transfer request status from all channels. Their bits are mirrored from the Channel Status Control registers (see [eTPU Channel n Status Control Register for Engine 1 \(eTPU_CCCR_CnSCR_ENG1\)](#)) and Host must write 1 to clear a status bit.

For details about interrupts, see [Interrupts and data transfer requests](#).

Address: 0h base + 210h offset = 210h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R																																
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

eTPU_GCR_CDTRSR_ENG1 field descriptions

Field	Description
0–31 DTRSn	Channel n Data Transfer Request Status. These bits mimic the corresponding eTPU DMA requests. DTRSn can be cleared by software writing a 1 to it or by the assertion of corresponding DMA completion acknowledge line. When read: 0.....Indicates that channel n has no pending data transfer request. 1.....Indicates that channel x has a pending data transfer request. When written: 0.....Keep status bit unaltered. 1.....Clear status bit.

37.12.4 eTPU Channel Data Transfer Request Status Register for Engine 2 (eTPU_GCR_CDTRSR_ENG2)

This register groups data transfer request status from all channels. These bits are mirrored from the Channel Status Control registers (see [eTPU Channel n Status Control Register for Engine 2 \(eTPU_CCCR_CnSCR_ENG2\)](#)) and Host must write 1 to clear a status bit.

For details about interrupts, see [Interrupts and data transfer requests](#).

Address: 0h base + 214h offset = 214h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	DTRSn																															
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

eTPU_GCR_CDTRSR_ENG2 field descriptions

Field	Description
0–31 DTRSn	Channel n Data Transfer Request Status. These bits mimic the corresponding eTPU DMA requests. DTRSn can be cleared by software writing a 1 to it or by the assertion of corresponding DMA completion acknowledge line. When read: 0.....Indicates that channel n has no pending data transfer request. 1.....Indicates that channel x has a pending data transfer request. When written: 0.....Keep status bit unaltered. 1.....Clear status bit.

37.12.5 eTPU Channel Interrupt Overflow Status Register for Engine 1 (eTPU_GCR_CIOSR_ENG1)

This register groups the interrupt overflow status from all channels. These bits are mirrored from the Channel Status Control registers (see [eTPU Channel n Status Control Register for Engine 1 \(eTPU_CCCR_CnSCR_ENG1\)](#)) and the Host must write 1 to clear a status bit.

For details about interrupts, see [Interrupts and data transfer requests](#).

Address: 0h base + 220h offset = 220h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	CIOSn																															
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

eTPU_GCR_CIOSR_ENG1 field descriptions

Field	Description
0–31 CIOSn	Channel n Interrupt Overflow Status When read: 0.....Indicates that no interrupt overflow occurred in the channel. 1.....Indicates that interrupt overflow occurred in the channel. When written: 0.....Keep status bit unaltered. 1.....Clear status bit.

37.12.6 eTPU Channel Interrupt Overflow Status Register for Engine 2 (eTPU_GCR_CIOSR_ENG2)

This register groups the interrupt overflow status from all channels. These bits are mirrored from the Channel Status Control registers (see [eTPU Channel n Status Control Register for Engine 2 \(eTPU_CCCR_CnSCR_ENG2\)](#)) and the Host must write 1 to clear a status bit.

For details about interrupts, see [Interrupts and data transfer requests](#).

Address: 0h base + 224h offset = 224h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	CIOSn																															
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	

eTPU_GCR_CIOSR_ENG2 field descriptions

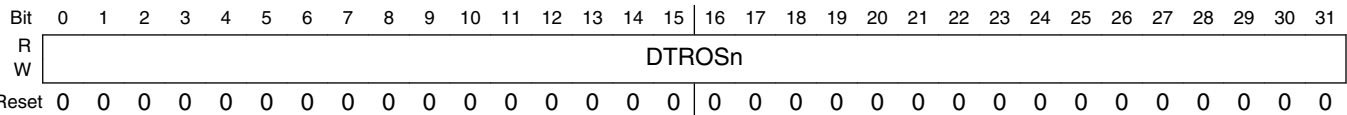
Field	Description
0–31 CIOSn	Channel n Interrupt Overflow Status When read: 0.....Indicates that no interrupt overflow occurred in the channel. 1.....Indicates that interrupt overflow occurred in the channel. When written: 0.....Keep status bit unaltered. 1.....Clear status bit.

37.12.7 eTPU Channel Data Transfer Request Overflow Status Register for Engine 1 (eTPU_GCR_CDTROSR_ENG1)

This register groups the data transfer request overflow status from all channels. These bits are mirrored from the Channel Status Control registers (see [eTPU Channel n Status Control Register for Engine 1 \(eTPU_CCCR_CnSCR_ENG1\)](#)) and the CPU must write 1 to clear a status bit.

For details about interrupts, see [Interrupts and data transfer requests](#).

Address: 0h base + 230h offset = 230h



eTPU_GCR_CDTROSR_ENG1 field descriptions

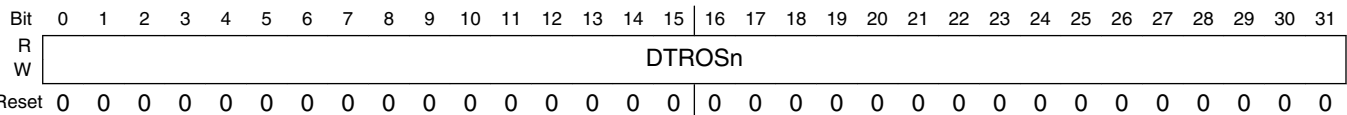
Field	Description
0–31 DTROSn	Channel n Data Transfer Request Overflow Status. When read: 0.....Indicates that no data transfer request overflow occurred in the channel. 1.....Indicates that data transfer request overflow occurred in the channel. When written: 0.....Keep status bit unaltered. 1.....Clear status bit.

37.12.8 eTPU Channel Data Transfer Request Overflow Status Register for Engine 2 (eTPU_GCR_CDTROSR_ENG2)

This register groups the data transfer request overflow status from all channels. These bits are mirrored from the Channel Status Control registers (see [eTPU Channel n Status Control Register for Engine 2 \(eTPU_CCCR_CnSCR_ENG2\)](#)) and the CPU must write 1 to clear a status bit.

For details about interrupts, see [Interrupts and data transfer requests](#).

Address: 0h base + 234h offset = 234h



eTPU_GCR_CDTROSR_ENG2 field descriptions

Field	Description
0–31 DTROSn	Channel n Data Transfer Request Overflow Status. When read: 0.....Indicates that no data transfer request overflow occurred in the channel. 1.....Indicates that data transfer request overflow occurred in the channel. When written: 0.....Keep status bit unaltered. 1.....Clear status bit.

37.12.9 eTPU Channel Interrupt Enable Register for Engine 1 (eTPU_GCR_CIER_ENG1)

This register groups the data transfer request overflow status from all channels. These bits are mirrored from the Channel Status Control registers (see [eTPU Channel n Status Control Register for Engine 1 \(eTPU_CCCR_CnSCR_ENG1\)](#)).

For details about interrupts, see [Interrupts and data transfer requests](#).

Address: 0h base + 240h offset = 240h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R																																
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

eTPU_GCR_CIER_ENG1 field descriptions

Field	Description
0–31 CIEn	Channel n Interrupt Enable. 0 Interrupt disabled for channel n. 1 Interrupt enabled for channel n.

37.12.10 eTPU Channel Interrupt Enable Register for Engine 2 (eTPU_GCR_CIER_ENG2)

This register groups the data transfer request overflow status from all channels. These bits are mirrored from the Channel Status Control registers (see [eTPU Channel n Status Control Register for Engine 2 \(eTPU_CCCR_CnSCR_ENG2\)](#)).

For details about interrupts, see [Interrupts and data transfer requests](#).

Global Channel memory map and registers

Address: 0h base + 244h offset = 244h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	CIEn																															
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	

eTPU_GCR_CIER_ENG2 field descriptions

Field	Description
0–31 CIEn	Channel n Interrupt Enable. 0 Interrupt disabled for channel n. 1 Interrupt enabled for channel n.

37.12.11 eTPU Channel Data Transfer Request Enable Register for Engine 1 (eTPU_GCR_CDTRER_ENG1)

This register groups the data transfer request enable from all channels. These bits are mirrored from the Channel Status Control registers (see [eTPU Channel n Status Control Register for Engine 1 \(eTPU_CCCR_CnSCR_ENG1\)](#)).

For details about interrupts, see [Interrupts and data transfer requests](#).

Address: 0h base + 250h offset = 250h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	DTREn																															
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	

eTPU_GCR_CDTRER_ENG1 field descriptions

Field	Description
0–31 DTREn	Channel n Data Transfer Request Enable. 0 Data Transfer request disabled for channel n. 1 Data Transfer request enabled for channel n.

37.12.12 eTPU Channel Data Transfer Request Enable Register for Engine 2 (eTPU_GCR_CDTRER_ENG2)

This register groups the data transfer request enable from all channels. These bits are mirrored from the Channel Status Control registers (see [eTPU Channel n Status Control Register for Engine 2 \(eTPU_CCCR_CnSCR_ENG2\)](#)).

For details about interrupts, see [Interrupts and data transfer requests](#).

Address: 0h base + 254h offset = 254h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	DTREn																															
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	

eTPU_GCR_CDTRER_ENG2 field descriptions

Field	Description
0–31 DTREn	Channel n Data Transfer Request Enable. 0 Data Transfer request disabled for channel n. 1 Data Transfer request enabled for channel n.

37.12.13 eTPU Watchdog Status Register for Engine 1 (eTPU_GCR_WDSR_ENG1)

This register indicates the watchdog influence on each of the engine channels.

Address: 0h base + 260h offset = 260h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	WDSn																															
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	

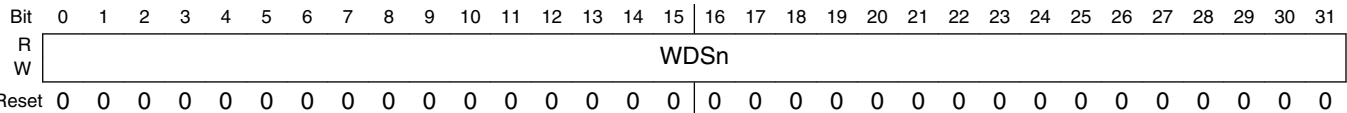
eTPU_GCR_WDSR_ENG1 field descriptions

Field	Description
0–31 WDSn	Channel n Watchdog Status. For details about Watchdog mechanism, see Watchdog . NOTE: The WDSn bits set in case of a watchdog timeout independently of WMO . When read: 0.....Watchdog did not expire during execution of a channel x thread since reset or last WDSn clear. 1.....Watchdog expired during execution of a channel n thread. When written: 0.....Keep watchdog status bit unaltered. 1.....Clear watchdog status bit.

37.12.14 eTPU Watchdog Status Register for Engine 2 (eTPU_GCR_WDSR_ENG2)

This register indicates the watchdog influence on each of the engine channels.

Address: 0h base + 264h offset = 264h



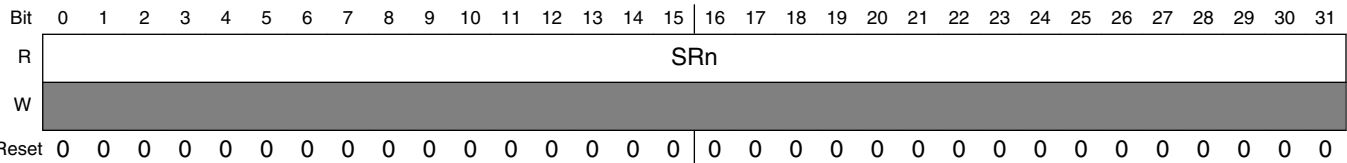
eTPU_GCR_WDSR_ENG2 field descriptions

Field	Description
0–31 WDSn	Channel n Watchdog Status. For details about Watchdog mechanism, see Watchdog. NOTE: The WDSn bits set in case of a watchdog timeout independently of WMO. When read: 0.....Watchdog did not expire during execution of a channel x thread since reset or last WDSn clear. 1.....Watchdog expired during execution of a channel n thread. When written: 0.....Keep watchdog status bit unaltered. 1.....Clear watchdog status bit.

37.12.15 eTPU Channel Pending Service Status Register for Engine 1 (eTPU_GCR_CPSSR_ENG1)

This register holds the status of the pending Channel Service Requests (see [Functions and threads](#)).

Address: 0h base + 280h offset = 280h



eTPU_GCR_CPSSR_ENG1 field descriptions

Field	Description
0–31 SRn	Pending Service Request n. Indicates a pending Service Request for channel n. Pending SR status is a logic OR of all service requests pending: if only HSR is active, SRn clears only at the end of the thread. SRn clear due to the other request sources is microcode dependent. NOTE: The pending service status bit for a channel is 1 when a Service Request is pending, even if the Channel is disabled (CPRn = 0). NOTE: There can be a delay of one clock between writing HSR greater than 0 in eTPU Channel n Host Service Request Register for Engine 1 (eTPU_CCCR_CnHSRR_ENG1) of a channel and its respective bit being asserted in eTPU_GCR_CPSSR_ENG1. 0 No Service Request pending for channel n. 1 Pending Service Request for channel n.

37.12.16 eTPU Channel Pending Service Status Register for Engine 2 (eTPU_GCR_CPSSR_ENG2)

This register is a read-only register that holds the status of the pending Channel Service Requests (see [Functions and threads](#)).

Address: 0h base + 284h offset = 284h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	SRn																															
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

eTPU_GCR_CPSSR_ENG2 field descriptions

Field	Description
0–31 SRn	Pending Service Request n. Indicates a pending Service Request for channel n. Pending SR status is a logic OR of all service requests pending: if only HSR is active, SRn clears only at the end of the thread. SRn clear due to the other request sources is microcode dependent. NOTE: The pending service status bit for a channel is 1 when a Service Request is pending, even if the Channel is disabled (CPRn = 0). NOTE: There can be a delay of one clock between writing HSR greater than 0 in eTPU Channel n Host Service Request Register for Engine 2 (eTPU_CCCR_CnHSRR_ENG2) of a channel and its respective bit being asserted in eTPU_GCR_CPSSR_ENG2. 0 No Service Request pending for channel n. 1 Pending Service Request for channel n.

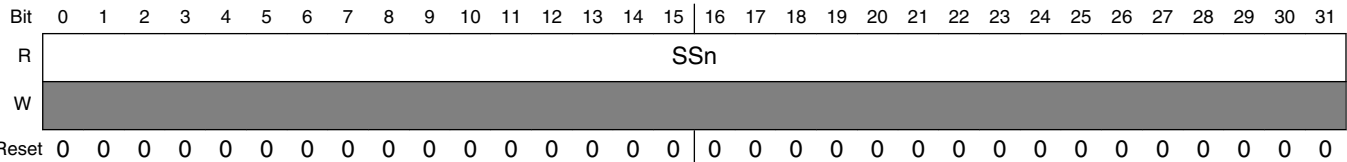
37.12.17 eTPU Channel Service Status Register for Engine 1
(eTPU_GCR_CSSR_ENG1)

This register holds the current channel service status on whether it is being serviced or not (see [Functions and threads](#)). Only one bit may be asserted in this register at a given time. When no channel is being serviced, the register read value is 0x0000_0000. This register is read-only and can be read during normal eTPU operation for monitoring the scheduler activity.

NOTE

Channel Service Status does not always reflect decoding of the CHAN register (see [CHAN Register](#)), since the latter can be changed by the service thread microcode.

Address: 0h base + 290h offset = 290h



eTPU_GCR_CSSR_ENG1 field descriptions

Field	Description
0–31 SSn	Service Status n. Indicates that channel n is currently being serviced. It is updated at the 1st microcycle of a Time Slot Transition (see Time Slot Transition), or when the microengine ends the thread. 0 Channel n is not currently being serviced. 1 channel n is currently being serviced.

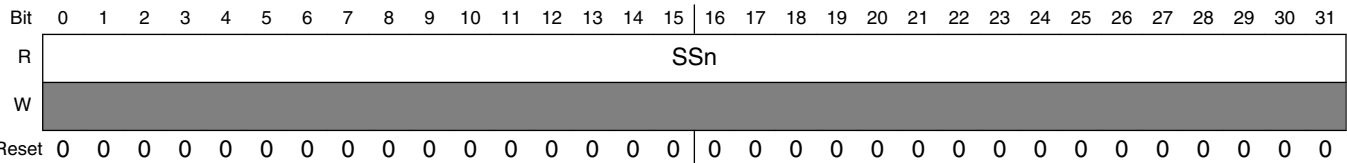
37.12.18 eTPU Channel Service Status Register for Engine 2
(eTPU_GCR_CSSR_ENG2)

This register holds the current channel service status on whether it is being serviced or not (see [Functions and threads](#)). Only one bit may be asserted in this register at a given time. When no channel is being serviced, the register read value is 0x0000_0000. This read-only register can be read during normal eTPU operation for monitoring the scheduler activity.

NOTE

Channel Service Status does not always reflect decoding of the CHAN register (see [CHAN Register](#)), since the latter can be changed by the service thread microcode.

Address: 0h base + 294h offset = 294h



eTPU_GCR_CSSR_ENG2 field descriptions

Field	Description
0–31 SSn	Service Status n. Indicates that channel n is currently being serviced. It is updated at the 1st microcycle of a Time Slot Transition (see Time Slot Transition), or when the microengine ends the thread. 0 Channel n is not currently being serviced. 1 channel n is currently being serviced.

37.13 Channel Configuration Control memory map and registers

Each channel has a group of 3 registers used to control, configure and check status of that channel as shown in the following table. This organization eases individual channel management.

Table 37-121. Channel registers structure

Channel offset	Register name
0x00	eTPU_CCCR_CnCR_ENGm - eTPU Channel Configuration Register
0x04	eTPU_CCCR_CnSCR_ENGm - eTPU Channel Status/Control Register
0x08	eTPU_CCCR_CnHSRR_ENGm - eTPU Channel Host Service Request Register
0x0C	Reserved

One contiguous area is used to map all channel registers of each eTPU engine as shown in the following table.

Table 37-122. Channel registers mMap

Offset	Registers structure
0x400	eTPU 1 Channel 0 Registers Structure
0x410	eTPU 1 Channel 1 Registers Structure
0x420	eTPU 1 Channel 2 Registers Structure
0x430	----
0x5E0	eTPU 1 Channel 30 Registers Structure
0x5F0	eTPU 1 Channel 31 Registers Structure
0x600	RESERVED
0x800	eTPU 2 Channel 0 Registers Structure
0x810	eTPU 2 Channel 1 Registers Structure
0x820	----
0x9E0	eTPU 2 Channel 30 Registers Structure
0x9F0	eTPU 2 Channel 31 Registers Structure
0xA00	RESERVED

There are 64 structures defined, one for each available channel in the eTPU System (32 for each Engine). The base address for the structure presented can be calculated by using the following equation:

$\text{Channel\_Register\_Base} = \text{eTPU\_Engine\_Channel\_Base} + (\text{channel\_number} * 0x10)$,
where:

- eTPU_Engine_Channel_Base = eTPU_Base + 0x400 for Engine 1.
- eTPU_Engine_Channel_Base = eTPU_Base + 0x800 for Engine 2.

eTPU_CCCR memory map

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/page
400	eTPU Channel n Configuration Register for Engine 1 (eTPU_CCCR_C0CR_ENG1)	32	R/W	0000_0000h	37.13.1/1585
404	eTPU Channel n Status Control Register for Engine 1 (eTPU_CCCR_C0SCR_ENG1)	32	R/W	See section	37.13.2/1588
408	eTPU Channel n Host Service Request Register for Engine 1 (eTPU_CCCR_C0HSRR_ENG1)	32	R/W	0000_0000h	37.13.3/1590
410	eTPU Channel n Configuration Register for Engine 1 (eTPU_CCCR_C1CR_ENG1)	32	R/W	0000_0000h	37.13.1/1585
414	eTPU Channel n Status Control Register for Engine 1 (eTPU_CCCR_C1SCR_ENG1)	32	R/W	See section	37.13.2/1588
418	eTPU Channel n Host Service Request Register for Engine 1 (eTPU_CCCR_C1HSRR_ENG1)	32	R/W	0000_0000h	37.13.3/1590

Table continues on the next page...

eTPU_CCCR memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
420	eTPU Channel n Configuration Register for Engine 1 (eTPU_CCCR_C2CR_ENG1)	32	R/W	0000_0000h	37.13.1/ 1585
424	eTPU Channel n Status Control Register for Engine 1 (eTPU_CCCR_C2SCR_ENG1)	32	R/W	See section	37.13.2/ 1588
428	eTPU Channel n Host Service Request Register for Engine 1 (eTPU_CCCR_C2HSRR_ENG1)	32	R/W	0000_0000h	37.13.3/ 1590
430	eTPU Channel n Configuration Register for Engine 1 (eTPU_CCCR_C3CR_ENG1)	32	R/W	0000_0000h	37.13.1/ 1585
434	eTPU Channel n Status Control Register for Engine 1 (eTPU_CCCR_C3SCR_ENG1)	32	R/W	See section	37.13.2/ 1588
438	eTPU Channel n Host Service Request Register for Engine 1 (eTPU_CCCR_C3HSRR_ENG1)	32	R/W	0000_0000h	37.13.3/ 1590
440	eTPU Channel n Configuration Register for Engine 1 (eTPU_CCCR_C4CR_ENG1)	32	R/W	0000_0000h	37.13.1/ 1585
444	eTPU Channel n Status Control Register for Engine 1 (eTPU_CCCR_C4SCR_ENG1)	32	R/W	See section	37.13.2/ 1588
448	eTPU Channel n Host Service Request Register for Engine 1 (eTPU_CCCR_C4HSRR_ENG1)	32	R/W	0000_0000h	37.13.3/ 1590
450	eTPU Channel n Configuration Register for Engine 1 (eTPU_CCCR_C5CR_ENG1)	32	R/W	0000_0000h	37.13.1/ 1585
454	eTPU Channel n Status Control Register for Engine 1 (eTPU_CCCR_C5SCR_ENG1)	32	R/W	See section	37.13.2/ 1588
458	eTPU Channel n Host Service Request Register for Engine 1 (eTPU_CCCR_C5HSRR_ENG1)	32	R/W	0000_0000h	37.13.3/ 1590
460	eTPU Channel n Configuration Register for Engine 1 (eTPU_CCCR_C6CR_ENG1)	32	R/W	0000_0000h	37.13.1/ 1585
464	eTPU Channel n Status Control Register for Engine 1 (eTPU_CCCR_C6SCR_ENG1)	32	R/W	See section	37.13.2/ 1588
468	eTPU Channel n Host Service Request Register for Engine 1 (eTPU_CCCR_C6HSRR_ENG1)	32	R/W	0000_0000h	37.13.3/ 1590
470	eTPU Channel n Configuration Register for Engine 1 (eTPU_CCCR_C7CR_ENG1)	32	R/W	0000_0000h	37.13.1/ 1585
474	eTPU Channel n Status Control Register for Engine 1 (eTPU_CCCR_C7SCR_ENG1)	32	R/W	See section	37.13.2/ 1588
478	eTPU Channel n Host Service Request Register for Engine 1 (eTPU_CCCR_C7HSRR_ENG1)	32	R/W	0000_0000h	37.13.3/ 1590
480	eTPU Channel n Configuration Register for Engine 1 (eTPU_CCCR_C8CR_ENG1)	32	R/W	0000_0000h	37.13.1/ 1585
484	eTPU Channel n Status Control Register for Engine 1 (eTPU_CCCR_C8SCR_ENG1)	32	R/W	See section	37.13.2/ 1588
488	eTPU Channel n Host Service Request Register for Engine 1 (eTPU_CCCR_C8HSRR_ENG1)	32	R/W	0000_0000h	37.13.3/ 1590
490	eTPU Channel n Configuration Register for Engine 1 (eTPU_CCCR_C9CR_ENG1)	32	R/W	0000_0000h	37.13.1/ 1585

Table continues on the next page...

eTPU_CCCR memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
494	eTPU Channel n Status Control Register for Engine 1 (eTPU_CCCR_C9SCR_ENG1)	32	R/W	See section	37.13.2/ 1588
498	eTPU Channel n Host Service Request Register for Engine 1 (eTPU_CCCR_C9HSRR_ENG1)	32	R/W	0000_0000h	37.13.3/ 1590
4A0	eTPU Channel n Configuration Register for Engine 1 (eTPU_CCCR_C10CR_ENG1)	32	R/W	0000_0000h	37.13.1/ 1585
4A4	eTPU Channel n Status Control Register for Engine 1 (eTPU_CCCR_C10SCR_ENG1)	32	R/W	See section	37.13.2/ 1588
4A8	eTPU Channel n Host Service Request Register for Engine 1 (eTPU_CCCR_C10HSRR_ENG1)	32	R/W	0000_0000h	37.13.3/ 1590
4B0	eTPU Channel n Configuration Register for Engine 1 (eTPU_CCCR_C11CR_ENG1)	32	R/W	0000_0000h	37.13.1/ 1585
4B4	eTPU Channel n Status Control Register for Engine 1 (eTPU_CCCR_C11SCR_ENG1)	32	R/W	See section	37.13.2/ 1588
4B8	eTPU Channel n Host Service Request Register for Engine 1 (eTPU_CCCR_C11HSRR_ENG1)	32	R/W	0000_0000h	37.13.3/ 1590
4C0	eTPU Channel n Configuration Register for Engine 1 (eTPU_CCCR_C12CR_ENG1)	32	R/W	0000_0000h	37.13.1/ 1585
4C4	eTPU Channel n Status Control Register for Engine 1 (eTPU_CCCR_C12SCR_ENG1)	32	R/W	See section	37.13.2/ 1588
4C8	eTPU Channel n Host Service Request Register for Engine 1 (eTPU_CCCR_C12HSRR_ENG1)	32	R/W	0000_0000h	37.13.3/ 1590
4D0	eTPU Channel n Configuration Register for Engine 1 (eTPU_CCCR_C13CR_ENG1)	32	R/W	0000_0000h	37.13.1/ 1585
4D4	eTPU Channel n Status Control Register for Engine 1 (eTPU_CCCR_C13SCR_ENG1)	32	R/W	See section	37.13.2/ 1588
4D8	eTPU Channel n Host Service Request Register for Engine 1 (eTPU_CCCR_C13HSRR_ENG1)	32	R/W	0000_0000h	37.13.3/ 1590
4E0	eTPU Channel n Configuration Register for Engine 1 (eTPU_CCCR_C14CR_ENG1)	32	R/W	0000_0000h	37.13.1/ 1585
4E4	eTPU Channel n Status Control Register for Engine 1 (eTPU_CCCR_C14SCR_ENG1)	32	R/W	See section	37.13.2/ 1588
4E8	eTPU Channel n Host Service Request Register for Engine 1 (eTPU_CCCR_C14HSRR_ENG1)	32	R/W	0000_0000h	37.13.3/ 1590
4F0	eTPU Channel n Configuration Register for Engine 1 (eTPU_CCCR_C15CR_ENG1)	32	R/W	0000_0000h	37.13.1/ 1585
4F4	eTPU Channel n Status Control Register for Engine 1 (eTPU_CCCR_C15SCR_ENG1)	32	R/W	See section	37.13.2/ 1588
4F8	eTPU Channel n Host Service Request Register for Engine 1 (eTPU_CCCR_C15HSRR_ENG1)	32	R/W	0000_0000h	37.13.3/ 1590
500	eTPU Channel n Configuration Register for Engine 1 (eTPU_CCCR_C16CR_ENG1)	32	R/W	0000_0000h	37.13.1/ 1585
504	eTPU Channel n Status Control Register for Engine 1 (eTPU_CCCR_C16SCR_ENG1)	32	R/W	See section	37.13.2/ 1588

Table continues on the next page...

eTPU_CCCR memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
508	eTPU Channel n Host Service Request Register for Engine 1 (eTPU_CCCR_C16HSRR_ENG1)	32	R/W	0000_0000h	37.13.3/ 1590
510	eTPU Channel n Configuration Register for Engine 1 (eTPU_CCCR_C17CR_ENG1)	32	R/W	0000_0000h	37.13.1/ 1585
514	eTPU Channel n Status Control Register for Engine 1 (eTPU_CCCR_C17SCR_ENG1)	32	R/W	See section	37.13.2/ 1588
518	eTPU Channel n Host Service Request Register for Engine 1 (eTPU_CCCR_C17HSRR_ENG1)	32	R/W	0000_0000h	37.13.3/ 1590
520	eTPU Channel n Configuration Register for Engine 1 (eTPU_CCCR_C18CR_ENG1)	32	R/W	0000_0000h	37.13.1/ 1585
524	eTPU Channel n Status Control Register for Engine 1 (eTPU_CCCR_C18SCR_ENG1)	32	R/W	See section	37.13.2/ 1588
528	eTPU Channel n Host Service Request Register for Engine 1 (eTPU_CCCR_C18HSRR_ENG1)	32	R/W	0000_0000h	37.13.3/ 1590
530	eTPU Channel n Configuration Register for Engine 1 (eTPU_CCCR_C19CR_ENG1)	32	R/W	0000_0000h	37.13.1/ 1585
534	eTPU Channel n Status Control Register for Engine 1 (eTPU_CCCR_C19SCR_ENG1)	32	R/W	See section	37.13.2/ 1588
538	eTPU Channel n Host Service Request Register for Engine 1 (eTPU_CCCR_C19HSRR_ENG1)	32	R/W	0000_0000h	37.13.3/ 1590
540	eTPU Channel n Configuration Register for Engine 1 (eTPU_CCCR_C20CR_ENG1)	32	R/W	0000_0000h	37.13.1/ 1585
544	eTPU Channel n Status Control Register for Engine 1 (eTPU_CCCR_C20SCR_ENG1)	32	R/W	See section	37.13.2/ 1588
548	eTPU Channel n Host Service Request Register for Engine 1 (eTPU_CCCR_C20HSRR_ENG1)	32	R/W	0000_0000h	37.13.3/ 1590
550	eTPU Channel n Configuration Register for Engine 1 (eTPU_CCCR_C21CR_ENG1)	32	R/W	0000_0000h	37.13.1/ 1585
554	eTPU Channel n Status Control Register for Engine 1 (eTPU_CCCR_C21SCR_ENG1)	32	R/W	See section	37.13.2/ 1588
558	eTPU Channel n Host Service Request Register for Engine 1 (eTPU_CCCR_C21HSRR_ENG1)	32	R/W	0000_0000h	37.13.3/ 1590
560	eTPU Channel n Configuration Register for Engine 1 (eTPU_CCCR_C22CR_ENG1)	32	R/W	0000_0000h	37.13.1/ 1585
564	eTPU Channel n Status Control Register for Engine 1 (eTPU_CCCR_C22SCR_ENG1)	32	R/W	See section	37.13.2/ 1588
568	eTPU Channel n Host Service Request Register for Engine 1 (eTPU_CCCR_C22HSRR_ENG1)	32	R/W	0000_0000h	37.13.3/ 1590
570	eTPU Channel n Configuration Register for Engine 1 (eTPU_CCCR_C23CR_ENG1)	32	R/W	0000_0000h	37.13.1/ 1585
574	eTPU Channel n Status Control Register for Engine 1 (eTPU_CCCR_C23SCR_ENG1)	32	R/W	See section	37.13.2/ 1588
578	eTPU Channel n Host Service Request Register for Engine 1 (eTPU_CCCR_C23HSRR_ENG1)	32	R/W	0000_0000h	37.13.3/ 1590

Table continues on the next page...

eTPU_CCCR memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
580	eTPU Channel n Configuration Register for Engine 1 (eTPU_CCCR_C24CR_ENG1)	32	R/W	0000_0000h	37.13.1/ 1585
584	eTPU Channel n Status Control Register for Engine 1 (eTPU_CCCR_C24SCR_ENG1)	32	R/W	See section	37.13.2/ 1588
588	eTPU Channel n Host Service Request Register for Engine 1 (eTPU_CCCR_C24HSRR_ENG1)	32	R/W	0000_0000h	37.13.3/ 1590
590	eTPU Channel n Configuration Register for Engine 1 (eTPU_CCCR_C25CR_ENG1)	32	R/W	0000_0000h	37.13.1/ 1585
594	eTPU Channel n Status Control Register for Engine 1 (eTPU_CCCR_C25SCR_ENG1)	32	R/W	See section	37.13.2/ 1588
598	eTPU Channel n Host Service Request Register for Engine 1 (eTPU_CCCR_C25HSRR_ENG1)	32	R/W	0000_0000h	37.13.3/ 1590
5A0	eTPU Channel n Configuration Register for Engine 1 (eTPU_CCCR_C26CR_ENG1)	32	R/W	0000_0000h	37.13.1/ 1585
5A4	eTPU Channel n Status Control Register for Engine 1 (eTPU_CCCR_C26SCR_ENG1)	32	R/W	See section	37.13.2/ 1588
5A8	eTPU Channel n Host Service Request Register for Engine 1 (eTPU_CCCR_C26HSRR_ENG1)	32	R/W	0000_0000h	37.13.3/ 1590
5B0	eTPU Channel n Configuration Register for Engine 1 (eTPU_CCCR_C27CR_ENG1)	32	R/W	0000_0000h	37.13.1/ 1585
5B4	eTPU Channel n Status Control Register for Engine 1 (eTPU_CCCR_C27SCR_ENG1)	32	R/W	See section	37.13.2/ 1588
5B8	eTPU Channel n Host Service Request Register for Engine 1 (eTPU_CCCR_C27HSRR_ENG1)	32	R/W	0000_0000h	37.13.3/ 1590
5C0	eTPU Channel n Configuration Register for Engine 1 (eTPU_CCCR_C28CR_ENG1)	32	R/W	0000_0000h	37.13.1/ 1585
5C4	eTPU Channel n Status Control Register for Engine 1 (eTPU_CCCR_C28SCR_ENG1)	32	R/W	See section	37.13.2/ 1588
5C8	eTPU Channel n Host Service Request Register for Engine 1 (eTPU_CCCR_C28HSRR_ENG1)	32	R/W	0000_0000h	37.13.3/ 1590
5D0	eTPU Channel n Configuration Register for Engine 1 (eTPU_CCCR_C29CR_ENG1)	32	R/W	0000_0000h	37.13.1/ 1585
5D4	eTPU Channel n Status Control Register for Engine 1 (eTPU_CCCR_C29SCR_ENG1)	32	R/W	See section	37.13.2/ 1588
5D8	eTPU Channel n Host Service Request Register for Engine 1 (eTPU_CCCR_C29HSRR_ENG1)	32	R/W	0000_0000h	37.13.3/ 1590
5E0	eTPU Channel n Configuration Register for Engine 1 (eTPU_CCCR_C30CR_ENG1)	32	R/W	0000_0000h	37.13.1/ 1585
5E4	eTPU Channel n Status Control Register for Engine 1 (eTPU_CCCR_C30SCR_ENG1)	32	R/W	See section	37.13.2/ 1588
5E8	eTPU Channel n Host Service Request Register for Engine 1 (eTPU_CCCR_C30HSRR_ENG1)	32	R/W	0000_0000h	37.13.3/ 1590
5F0	eTPU Channel n Configuration Register for Engine 1 (eTPU_CCCR_C31CR_ENG1)	32	R/W	0000_0000h	37.13.1/ 1585

Table continues on the next page...

eTPU_CCCR memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
5F4	eTPU Channel n Status Control Register for Engine 1 (eTPU_CCCR_C31SCR_ENG1)	32	R/W	See section	37.13.2/ 1588
5F8	eTPU Channel n Host Service Request Register for Engine 1 (eTPU_CCCR_C31HSRR_ENG1)	32	R/W	0000_0000h	37.13.3/ 1590
800	eTPU Channel n Configuration Register for Engine 2 (eTPU_CCCR_C0CR_ENG2)	32	R/W	0000_0000h	37.13.4/ 1591
804	eTPU Channel n Status Control Register for Engine 2 (eTPU_CCCR_C0SCR_ENG2)	32	R/W	See section	37.13.5/ 1594
808	eTPU Channel n Host Service Request Register for Engine 2 (eTPU_CCCR_C0HSRR_ENG2)	32	R/W	0000_0000h	37.13.6/ 1596
810	eTPU Channel n Configuration Register for Engine 2 (eTPU_CCCR_C1CR_ENG2)	32	R/W	0000_0000h	37.13.4/ 1591
814	eTPU Channel n Status Control Register for Engine 2 (eTPU_CCCR_C1SCR_ENG2)	32	R/W	See section	37.13.5/ 1594
818	eTPU Channel n Host Service Request Register for Engine 2 (eTPU_CCCR_C1HSRR_ENG2)	32	R/W	0000_0000h	37.13.6/ 1596
820	eTPU Channel n Configuration Register for Engine 2 (eTPU_CCCR_C2CR_ENG2)	32	R/W	0000_0000h	37.13.4/ 1591
824	eTPU Channel n Status Control Register for Engine 2 (eTPU_CCCR_C2SCR_ENG2)	32	R/W	See section	37.13.5/ 1594
828	eTPU Channel n Host Service Request Register for Engine 2 (eTPU_CCCR_C2HSRR_ENG2)	32	R/W	0000_0000h	37.13.6/ 1596
830	eTPU Channel n Configuration Register for Engine 2 (eTPU_CCCR_C3CR_ENG2)	32	R/W	0000_0000h	37.13.4/ 1591
834	eTPU Channel n Status Control Register for Engine 2 (eTPU_CCCR_C3SCR_ENG2)	32	R/W	See section	37.13.5/ 1594
838	eTPU Channel n Host Service Request Register for Engine 2 (eTPU_CCCR_C3HSRR_ENG2)	32	R/W	0000_0000h	37.13.6/ 1596
840	eTPU Channel n Configuration Register for Engine 2 (eTPU_CCCR_C4CR_ENG2)	32	R/W	0000_0000h	37.13.4/ 1591
844	eTPU Channel n Status Control Register for Engine 2 (eTPU_CCCR_C4SCR_ENG2)	32	R/W	See section	37.13.5/ 1594
848	eTPU Channel n Host Service Request Register for Engine 2 (eTPU_CCCR_C4HSRR_ENG2)	32	R/W	0000_0000h	37.13.6/ 1596
850	eTPU Channel n Configuration Register for Engine 2 (eTPU_CCCR_C5CR_ENG2)	32	R/W	0000_0000h	37.13.4/ 1591
854	eTPU Channel n Status Control Register for Engine 2 (eTPU_CCCR_C5SCR_ENG2)	32	R/W	See section	37.13.5/ 1594
858	eTPU Channel n Host Service Request Register for Engine 2 (eTPU_CCCR_C5HSRR_ENG2)	32	R/W	0000_0000h	37.13.6/ 1596
860	eTPU Channel n Configuration Register for Engine 2 (eTPU_CCCR_C6CR_ENG2)	32	R/W	0000_0000h	37.13.4/ 1591
864	eTPU Channel n Status Control Register for Engine 2 (eTPU_CCCR_C6SCR_ENG2)	32	R/W	See section	37.13.5/ 1594

Table continues on the next page...

eTPU_CCCR memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
868	eTPU Channel n Host Service Request Register for Engine 2 (eTPU_CCCR_C6HSRR_ENG2)	32	R/W	0000_0000h	37.13.6/1596
870	eTPU Channel n Configuration Register for Engine 2 (eTPU_CCCR_C7CR_ENG2)	32	R/W	0000_0000h	37.13.4/1591
874	eTPU Channel n Status Control Register for Engine 2 (eTPU_CCCR_C7SCR_ENG2)	32	R/W	See section	37.13.5/1594
878	eTPU Channel n Host Service Request Register for Engine 2 (eTPU_CCCR_C7HSRR_ENG2)	32	R/W	0000_0000h	37.13.6/1596
880	eTPU Channel n Configuration Register for Engine 2 (eTPU_CCCR_C8CR_ENG2)	32	R/W	0000_0000h	37.13.4/1591
884	eTPU Channel n Status Control Register for Engine 2 (eTPU_CCCR_C8SCR_ENG2)	32	R/W	See section	37.13.5/1594
888	eTPU Channel n Host Service Request Register for Engine 2 (eTPU_CCCR_C8HSRR_ENG2)	32	R/W	0000_0000h	37.13.6/1596
890	eTPU Channel n Configuration Register for Engine 2 (eTPU_CCCR_C9CR_ENG2)	32	R/W	0000_0000h	37.13.4/1591
894	eTPU Channel n Status Control Register for Engine 2 (eTPU_CCCR_C9SCR_ENG2)	32	R/W	See section	37.13.5/1594
898	eTPU Channel n Host Service Request Register for Engine 2 (eTPU_CCCR_C9HSRR_ENG2)	32	R/W	0000_0000h	37.13.6/1596
8A0	eTPU Channel n Configuration Register for Engine 2 (eTPU_CCCR_C10CR_ENG2)	32	R/W	0000_0000h	37.13.4/1591
8A4	eTPU Channel n Status Control Register for Engine 2 (eTPU_CCCR_C10SCR_ENG2)	32	R/W	See section	37.13.5/1594
8A8	eTPU Channel n Host Service Request Register for Engine 2 (eTPU_CCCR_C10HSRR_ENG2)	32	R/W	0000_0000h	37.13.6/1596
8B0	eTPU Channel n Configuration Register for Engine 2 (eTPU_CCCR_C11CR_ENG2)	32	R/W	0000_0000h	37.13.4/1591
8B4	eTPU Channel n Status Control Register for Engine 2 (eTPU_CCCR_C11SCR_ENG2)	32	R/W	See section	37.13.5/1594
8B8	eTPU Channel n Host Service Request Register for Engine 2 (eTPU_CCCR_C11HSRR_ENG2)	32	R/W	0000_0000h	37.13.6/1596
8C0	eTPU Channel n Configuration Register for Engine 2 (eTPU_CCCR_C12CR_ENG2)	32	R/W	0000_0000h	37.13.4/1591
8C4	eTPU Channel n Status Control Register for Engine 2 (eTPU_CCCR_C12SCR_ENG2)	32	R/W	See section	37.13.5/1594
8C8	eTPU Channel n Host Service Request Register for Engine 2 (eTPU_CCCR_C12HSRR_ENG2)	32	R/W	0000_0000h	37.13.6/1596
8D0	eTPU Channel n Configuration Register for Engine 2 (eTPU_CCCR_C13CR_ENG2)	32	R/W	0000_0000h	37.13.4/1591
8D4	eTPU Channel n Status Control Register for Engine 2 (eTPU_CCCR_C13SCR_ENG2)	32	R/W	See section	37.13.5/1594
8D8	eTPU Channel n Host Service Request Register for Engine 2 (eTPU_CCCR_C13HSRR_ENG2)	32	R/W	0000_0000h	37.13.6/1596

Table continues on the next page...

eTPU_CCCR memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
8E0	eTPU Channel n Configuration Register for Engine 2 (eTPU_CCCR_C14CR_ENG2)	32	R/W	0000_0000h	37.13.4/1591
8E4	eTPU Channel n Status Control Register for Engine 2 (eTPU_CCCR_C14SCR_ENG2)	32	R/W	See section	37.13.5/1594
8E8	eTPU Channel n Host Service Request Register for Engine 2 (eTPU_CCCR_C14HSRR_ENG2)	32	R/W	0000_0000h	37.13.6/1596
8F0	eTPU Channel n Configuration Register for Engine 2 (eTPU_CCCR_C15CR_ENG2)	32	R/W	0000_0000h	37.13.4/1591
8F4	eTPU Channel n Status Control Register for Engine 2 (eTPU_CCCR_C15SCR_ENG2)	32	R/W	See section	37.13.5/1594
8F8	eTPU Channel n Host Service Request Register for Engine 2 (eTPU_CCCR_C15HSRR_ENG2)	32	R/W	0000_0000h	37.13.6/1596
900	eTPU Channel n Configuration Register for Engine 2 (eTPU_CCCR_C16CR_ENG2)	32	R/W	0000_0000h	37.13.4/1591
904	eTPU Channel n Status Control Register for Engine 2 (eTPU_CCCR_C16SCR_ENG2)	32	R/W	See section	37.13.5/1594
908	eTPU Channel n Host Service Request Register for Engine 2 (eTPU_CCCR_C16HSRR_ENG2)	32	R/W	0000_0000h	37.13.6/1596
910	eTPU Channel n Configuration Register for Engine 2 (eTPU_CCCR_C17CR_ENG2)	32	R/W	0000_0000h	37.13.4/1591
914	eTPU Channel n Status Control Register for Engine 2 (eTPU_CCCR_C17SCR_ENG2)	32	R/W	See section	37.13.5/1594
918	eTPU Channel n Host Service Request Register for Engine 2 (eTPU_CCCR_C17HSRR_ENG2)	32	R/W	0000_0000h	37.13.6/1596
920	eTPU Channel n Configuration Register for Engine 2 (eTPU_CCCR_C18CR_ENG2)	32	R/W	0000_0000h	37.13.4/1591
924	eTPU Channel n Status Control Register for Engine 2 (eTPU_CCCR_C18SCR_ENG2)	32	R/W	See section	37.13.5/1594
928	eTPU Channel n Host Service Request Register for Engine 2 (eTPU_CCCR_C18HSRR_ENG2)	32	R/W	0000_0000h	37.13.6/1596
930	eTPU Channel n Configuration Register for Engine 2 (eTPU_CCCR_C19CR_ENG2)	32	R/W	0000_0000h	37.13.4/1591
934	eTPU Channel n Status Control Register for Engine 2 (eTPU_CCCR_C19SCR_ENG2)	32	R/W	See section	37.13.5/1594
938	eTPU Channel n Host Service Request Register for Engine 2 (eTPU_CCCR_C19HSRR_ENG2)	32	R/W	0000_0000h	37.13.6/1596
940	eTPU Channel n Configuration Register for Engine 2 (eTPU_CCCR_C20CR_ENG2)	32	R/W	0000_0000h	37.13.4/1591
944	eTPU Channel n Status Control Register for Engine 2 (eTPU_CCCR_C20SCR_ENG2)	32	R/W	See section	37.13.5/1594
948	eTPU Channel n Host Service Request Register for Engine 2 (eTPU_CCCR_C20HSRR_ENG2)	32	R/W	0000_0000h	37.13.6/1596
950	eTPU Channel n Configuration Register for Engine 2 (eTPU_CCCR_C21CR_ENG2)	32	R/W	0000_0000h	37.13.4/1591

Table continues on the next page...

eTPU_CCCR memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
954	eTPU Channel n Status Control Register for Engine 2 (eTPU_CCCR_C21SCR_ENG2)	32	R/W	See section	37.13.5/1594
958	eTPU Channel n Host Service Request Register for Engine 2 (eTPU_CCCR_C21HSRR_ENG2)	32	R/W	0000_0000h	37.13.6/1596
960	eTPU Channel n Configuration Register for Engine 2 (eTPU_CCCR_C22CR_ENG2)	32	R/W	0000_0000h	37.13.4/1591
964	eTPU Channel n Status Control Register for Engine 2 (eTPU_CCCR_C22SCR_ENG2)	32	R/W	See section	37.13.5/1594
968	eTPU Channel n Host Service Request Register for Engine 2 (eTPU_CCCR_C22HSRR_ENG2)	32	R/W	0000_0000h	37.13.6/1596
970	eTPU Channel n Configuration Register for Engine 2 (eTPU_CCCR_C23CR_ENG2)	32	R/W	0000_0000h	37.13.4/1591
974	eTPU Channel n Status Control Register for Engine 2 (eTPU_CCCR_C23SCR_ENG2)	32	R/W	See section	37.13.5/1594
978	eTPU Channel n Host Service Request Register for Engine 2 (eTPU_CCCR_C23HSRR_ENG2)	32	R/W	0000_0000h	37.13.6/1596
980	eTPU Channel n Configuration Register for Engine 2 (eTPU_CCCR_C24CR_ENG2)	32	R/W	0000_0000h	37.13.4/1591
984	eTPU Channel n Status Control Register for Engine 2 (eTPU_CCCR_C24SCR_ENG2)	32	R/W	See section	37.13.5/1594
988	eTPU Channel n Host Service Request Register for Engine 2 (eTPU_CCCR_C24HSRR_ENG2)	32	R/W	0000_0000h	37.13.6/1596
990	eTPU Channel n Configuration Register for Engine 2 (eTPU_CCCR_C25CR_ENG2)	32	R/W	0000_0000h	37.13.4/1591
994	eTPU Channel n Status Control Register for Engine 2 (eTPU_CCCR_C25SCR_ENG2)	32	R/W	See section	37.13.5/1594
998	eTPU Channel n Host Service Request Register for Engine 2 (eTPU_CCCR_C25HSRR_ENG2)	32	R/W	0000_0000h	37.13.6/1596
9A0	eTPU Channel n Configuration Register for Engine 2 (eTPU_CCCR_C26CR_ENG2)	32	R/W	0000_0000h	37.13.4/1591
9A4	eTPU Channel n Status Control Register for Engine 2 (eTPU_CCCR_C26SCR_ENG2)	32	R/W	See section	37.13.5/1594
9A8	eTPU Channel n Host Service Request Register for Engine 2 (eTPU_CCCR_C26HSRR_ENG2)	32	R/W	0000_0000h	37.13.6/1596
9B0	eTPU Channel n Configuration Register for Engine 2 (eTPU_CCCR_C27CR_ENG2)	32	R/W	0000_0000h	37.13.4/1591
9B4	eTPU Channel n Status Control Register for Engine 2 (eTPU_CCCR_C27SCR_ENG2)	32	R/W	See section	37.13.5/1594
9B8	eTPU Channel n Host Service Request Register for Engine 2 (eTPU_CCCR_C27HSRR_ENG2)	32	R/W	0000_0000h	37.13.6/1596
9C0	eTPU Channel n Configuration Register for Engine 2 (eTPU_CCCR_C28CR_ENG2)	32	R/W	0000_0000h	37.13.4/1591
9C4	eTPU Channel n Status Control Register for Engine 2 (eTPU_CCCR_C28SCR_ENG2)	32	R/W	See section	37.13.5/1594

Table continues on the next page...

eTPU_CCCR memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
9C8	eTPU Channel n Host Service Request Register for Engine 2 (eTPU_CCCR_C28HSRR_ENG2)	32	R/W	0000_0000h	37.13.6/1596
9D0	eTPU Channel n Configuration Register for Engine 2 (eTPU_CCCR_C29CR_ENG2)	32	R/W	0000_0000h	37.13.4/1591
9D4	eTPU Channel n Status Control Register for Engine 2 (eTPU_CCCR_C29SCR_ENG2)	32	R/W	See section	37.13.5/1594
9D8	eTPU Channel n Host Service Request Register for Engine 2 (eTPU_CCCR_C29HSRR_ENG2)	32	R/W	0000_0000h	37.13.6/1596
9E0	eTPU Channel n Configuration Register for Engine 2 (eTPU_CCCR_C30CR_ENG2)	32	R/W	0000_0000h	37.13.4/1591
9E4	eTPU Channel n Status Control Register for Engine 2 (eTPU_CCCR_C30SCR_ENG2)	32	R/W	See section	37.13.5/1594
9E8	eTPU Channel n Host Service Request Register for Engine 2 (eTPU_CCCR_C30HSRR_ENG2)	32	R/W	0000_0000h	37.13.6/1596
9F0	eTPU Channel n Configuration Register for Engine 2 (eTPU_CCCR_C31CR_ENG2)	32	R/W	0000_0000h	37.13.4/1591
9F4	eTPU Channel n Status Control Register for Engine 2 (eTPU_CCCR_C31SCR_ENG2)	32	R/W	See section	37.13.5/1594
9F8	eTPU Channel n Host Service Request Register for Engine 2 (eTPU_CCCR_C31HSRR_ENG2)	32	R/W	0000_0000h	37.13.6/1596

37.13.1 eTPU Channel n Configuration Register for Engine 1 (eTPU_CCCR_CnCR_ENG1)

This register gathers configurations set individually per channel.

Address: 0h base + 400h offset + (16d × i), where i=0d to 31d

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R										0						
W	CIE	DTRE	CPR	T4DIS	SSMAE	ETPD	ETCS									
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R																
W	ODIS	OPOL														
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

eTPU_CCCR_CnCR_ENG1 field descriptions

Field	Description
0 CIE	Channel Interrupt Enable (see Channel Interrupt and Data Transfer Requests). This bit is mirrored from eTPU Channel Interrupt Enable Register for Engine 1 (eTPU_GCR_CIER_ENG1). 0 Disable interrupt for this channel. 1 Enable interrupt for this channel.
1 DTRE	Channel Data Transfer Request Enable (see Interrupts and data transfer requests). This bit is mirrored from eTPU Channel Data Transfer Request Enable Register for Engine 1 (eTPU_GCR_CDTRER_ENG1). 0 Disable data transfer request for this channel. 1 Enable data transfer request for this channel.
2–3 CPR	Channel Priority. This field defines the priority level for the channel, used by the Hardware Scheduler (see Scheduler). 00 Disabled 01 Low 10 Middle 11 High
4 T4DIS	T4 channel action Disable. This bit forces the channel to work on T2 timing mode, regardless of the FCSS and CDFC fields in eTPU Engine 1 Configuration Register (eTPU_SCR_ECR_ENG1), or of the TCR1CS field in eTPU Engine 1 Time Base Configuration Register (eTPU_TBR_TBCR_ENG1). For more details, see Channel timing modes. 0 Channel works either in T2 or T2/T4 timing modes, depending on the TCR1CS field in eTPU Engine 1 Time Base Configuration Register (eTPU_TBR_TBCR_ENG1) and the FCSS and CDFC fields in eTPU Engine 1 Configuration Register (eTPU_SCR_ECR_ENG1). 1 Channel works in T2 timing mode, regardless of the TCR1CS field in eTPU Engine 1 Time Base Configuration Register (eTPU_TBR_TBCR_ENG1) or the FCSS and CDFC fields in eTPU Engine 1 Configuration Register (eTPU_SCR_ECR_ENG1).
5 SSMAE	Shared Subsystem Microengine Access Enable. This bit enables the microengine access to the Shared Subsystem Area (see Shared Subsystems support). When this bit is negated, DIOB address values pointing to the Shared Subsystem Area actually access SDM mirrors, in legacy mode. NOTE: SSMAE value must only be changed when the engine is either stopped or halted. 0 An SDM mirror is accessed when the Shared Subsystem Area is addressed by the microengine. 1 Enables microengine access to the Shared Subsystem Area.
6 ETPD	Entry Table Pin Direction. This bit selects which channel signal, input or output, is used in the Entry Point selection. The ETPD value has to be compatible with the function chosen for the channel, selected in the field CFS. For details about Entry Table and condition encoding schemes, see Entry Points.

Table continues on the next page...

eTPU_CCCR_CnCR_ENG1 field descriptions (continued)

Field	Description
	0 Use PSTI for Entry Point selection. 1 Use PSTO for Entry Point selection.
7 ETCS	Entry Table Condition Select. This bit determines the channel condition encoding scheme that selects, according to channel conditions, the Entry Point to be taken in an Entry Table. ETCS value has to be compatible with the function chosen for the channel, selected in field CFS. Two condition encoding schemes are available. For details about Entry Table and condition encoding schemes, see Entry Points . 0 Select Standard Entry Table Condition encoding scheme. 1 Select Alternate Entry Table Condition encoding scheme.
8–10 Reserved	Reserved. This field is reserved. This read-only field is reserved and always has the value 0.
11–15 CFS	Channel Function Select. This field defines the function to be performed by the channel (see Functions and threads). The Function assigned to the channel has to be compatible with the channel condition encoding scheme, selected by field ETCS.
16 ODIS	Output Disable. This bit enables the channel to have its output forced to the value opposite to OPOL when the output disable input signal corresponding to the channel group that it belongs is active (see eTPU channel output disable signals and Figure 37-305). 0 Turns off the output disable feature for the channel. 1 Turns on the output disable feature for the channel.
17 OPOL	Output Polarity. Determines the output signal polarity. The activation of the output disable signal forces, when enabled by the ODIS bit, the channel output signal to the opposite of this polarity (see Figure 37-305). 0 Output active low (output disable drives output to high). 1 output active high (output disable drives output to low).
18–20 Reserved	Reserved. This field is reserved. This read-only field is reserved and always has the value 0.
21–31 CPBA	Channel n Parameter Base Address. The value of this field times 8 specifies the SDM parameter base host (byte) address for channel n (2-parameter granularity; see SDM organization). As seen by the Host, the channel parameter base (byte) address is (without parameter sign extension) $eTPU\_Base + 0x8000 + CPBA * 8$ or (with parameter sign extension), $eTPU\_Base + 0xC000 + CPBA * 8$.

37.13.2 eTPU Channel n Status Control Register for Engine 1 (eTPU_CCCR_CnSCR_ENG1)

This register gathers the interrupt status bits of the channel and the Function Mode definition (read-write). Bits CIS, CIOS and DTRS for each channel can be also accessed from [eTPU Channel Interrupt Status Register for Engine 1 \(eTPU_GCR_CISR_ENG1\)](#), [eTPU Channel Interrupt Overflow Status Register for Engine 1 \(eTPU_GCR_CIOSR_ENG1\)](#) and [eTPU Channel Data Transfer Request Status Register for Engine 1 \(eTPU_GCR_CDTRSR_ENG1\)](#). Host must write 1 to clear a status bit.

Address: 0h base + 404h offset + (16d × i), where i=0d to 31d

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	CIS	CLOS	0						DTRS	DTROS	0					
W	w1c	w1c							w1c	w1c						
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	IPS	OPS	0	0												FM
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

eTPU_CCCR_CnSCR_ENG1 field descriptions

Field	Description
0 CIS	Channel Interrupt Status. These bits are mirrored in eTPU Channel Interrupt Status Register for Engine 1 (eTPU_GCR_CISR_ENG1) . Also, see Channel Interrupt and Data Transfer Requests . When read: 0.....Channel has no pending interrupt to the Host CPU. 1.....Channel has a pending interrupt to the Host CPU. When written: 0.....Keep interrupt status bit unaltered.

Table continues on the next page...

eTPU_CCCR_CnSCR_ENG1 field descriptions (continued)

Field	Description
	1.....Clear interrupt status bit.
1 CIOS	Channel Interrupt Overflow Status. These bits are mirrored in eTPU Channel Interrupt Overflow Status Register for Engine 1 (eTPU_GCR_CIOSR_ENG1). Also, see Interrupt and Data Transfer Request overflow. When read: 0.....Interrupt overflow negated for this channel. 1.....Interrupt overflow asserted for this channel. When written: 0.....Keep status bit unaltered. 1.....Clear status bit.
2–7 Reserved	Reserved. This field is reserved. This read-only field is reserved and always has the value 0.
8 DTRS	Data Transfer Request Status. These bits are mirrored in eTPU Channel Data Transfer Request Status Register for Engine 1 (eTPU_GCR_CDTRSR_ENG1). Also, see Channel Interrupt and Data Transfer Requests. When read: 0.....Channel has no pending data transfer request. 1.....Channel has a pending data transfer request. When written: 0.....Keep status bit unaltered. 1.....Clear status bit.
9 DTROS	Data Transfer Request Overflow Status. These bits are mirrored in eTPU Channel Data Transfer Request Overflow Status Register for Engine 1 (eTPU_GCR_CDTROSR_ENG1). Also, see Interrupt and Data Transfer Request overflow. When read: 0.....Data transfer request overflow negated for this channel. 1.....Data transfer request overflow asserted for this channel. When written: 0.....Keep status bit unaltered. 1.....Clear status bit.
10–15 Reserved	Reserved. This field is reserved. This read-only field is reserved and always has the value 0.
16 IPS	Channel Input Pin State. This bit shows the current value of the filtered channel input signal state.
17 OPS	Channel Output Pin State.

Table continues on the next page...

eTPU_CCCR_CnSCR_ENG1 field descriptions (continued)

Field	Description
	This bit shows the current value driven in the channel output signal, including the effect of the external output disable feature (see eTPU channel output disable signals). If the channel input and output signals are connected to the same pad, OPS reflects the value driven to the pad. This is not necessarily the actual pad value, which drives the value in the bit IPS.
18 Reserved	Reserved. This field is reserved. This read-only field is reserved and always has the value 0.
19–29 Reserved	Reserved. This field is reserved. This read-only field is reserved and always has the value 0.
30–31 FM	Channel Function Mode. Each function uses this field for a specific configuration. These bits can be tested by microengine code (see Conditional/unconditional branch). These bits are equivalent to the TPU/TPU2/TPU3 Host Sequence (HSQ) bits.

37.13.3 eTPU Channel n Host Service Request Register for Engine 1 (eTPU_CCCR_CnHSRR_ENG1)

This register is used by the Host to issue service requests to the channel.

Address: 0h base + 408h offset + (16d × i), where i=0d to 31d

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0																												HSR			
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	

eTPU_CCCR_CnHSRR_ENG1 field descriptions

Field	Description
0–28 Reserved	Reserved. This field is reserved. This read-only field is reserved and always has the value 0.
29–31 HSR	Host Service Request This field is used by the Host CPU to request service to the channel (see Host Service Requests). HSR value turns to 000 automatically at the end of microengine service for that channel, but only if the thread started due to an HSR. Host should write HSR greater than 0 only when HSR=0. Writing HSR=000 withdraws a pending request if scheduler did not begin to resolve the Entry Point yet, but it does not abort the service thread from that point on. For more details, see Entry Points and Host Service Requests . 000 No Host Service Request pending.

Table continues on the next page...

eTPU_CCCR_CnHSRR_ENG1 field descriptions (continued)

Field	Description
001	Function-dependent Host Service Request pending.
---	---
111	Function-dependent Host Service Request pending.

37.13.4 eTPU Channel n Configuration Register for Engine 2 (eTPU_CCCR_CnCR_ENG2)

This register gathers configurations set individually per channel.

Address: 0h base + 800h offset + (16d × i), where i=0d to 31d

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R										0						
W	CIE	DTRE	CPR		T4DIS	SSMAE	ETPD	ETCS								
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R																
W	ODIS	OPOL														
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

eTPU_CCCR_CnCR_ENG2 field descriptions

Field	Description
0 CIE	Channel Interrupt Enable (see Channel Interrupt and Data Transfer Requests). This bit is mirrored from eTPU Channel Interrupt Enable Register for Engine 2 (eTPU_GCR_CIER_ENG2) . 0 Disable interrupt for this channel. 1 Enable interrupt for this channel.
1 DTRE	Channel Data Transfer Request Enable (see Channel Interrupt and Data Transfer Requests). This bit is mirrored from eTPU Channel Data Transfer Request Enable Register for Engine 2 (eTPU_GCR_CDTRE_ENG2) . 0 Disable data transfer request for this channel. 1 Enable data transfer request for this channel.

Table continues on the next page...

eTPU_CCCR_CnCR_ENG2 field descriptions (continued)

Field	Description
2–3 CPR	Channel Priority. This field defines the priority level for the channel, used by the Hardware Scheduler (see Scheduler). 00 Disabled 01 Low 10 Middle 11 High
4 T4DIS	T4 channel action Disable. This bit forces the channel to work on T2 timing mode, regardless of the FCSS and CDFC fields in eTPU Engine 2 Configuration Register (eTPU_SCR_ECR_ENG2), or of the TCR1CS field in eTPU Engine 2 Time Base Configuration Register (eTPU_TBR_TBCR_ENG2). For more details, see Channel timing modes. 0 Channel works either in T2 or T2/T4 timing modes, depending on the TCR1CS field in eTPU Engine 2 Time Base Configuration Register (eTPU_TBR_TBCR_ENG2) and the FCSS and CDFC fields in eTPU Engine 2 Configuration Register (eTPU_SCR_ECR_ENG2). 1 Channel works in T2 timing mode, regardless of the TCR1CS field in eTPU Engine 2 Time Base Configuration Register (eTPU_TBR_TBCR_ENG2) or the FCSS and CDFC fields in eTPU Engine 2 Configuration Register (eTPU_SCR_ECR_ENG2).
5 SSMAE	Shared Subsystem Microengine Access Enable. This bit enables the microengine access to the Shared Subsystem Area (see Shared Subsystems support). When this bit is negated, DIOB address values pointing to the Shared Subsystem Area actually access SDM mirrors, in legacy mode. NOTE: SSMAE value must only be changed when the engine is either stopped or halted. 0 An SDM mirror is accessed when the Shared Subsystem Area is addressed by the microengine. 1 Enables microengine access to the Shared Subsystem Area.
6 ETPD	Entry Table Pin Direction. This bit selects which channel signal, input or output, is used in the Entry Point selection. The ETPD value has to be compatible with the function chosen for the channel, selected in the field CFS. For details about Entry Table and condition encoding schemes, see Entry Points. 0 Use PSTI for Entry Point selection. 1 Use PSTO for Entry Point selection.
7 ETCS	Entry Table Condition Select. This bit determines the channel condition encoding scheme that selects, according to channel conditions, the Entry Point to be taken in an Entry Table. ETCS value has to be compatible with the function chosen for the channel, selected in field CFS. Two condition encoding schemes are available. For details about Entry Table and condition encoding schemes, see Entry Points. 0 Select Standard Entry Table Condition encoding scheme. 1 Select Alternate Entry Table Condition encoding scheme.
8–10 Reserved	Reserved. This field is reserved. This read-only field is reserved and always has the value 0.

Table continues on the next page...

eTPU_CCCR_CnCR_ENG2 field descriptions (continued)

Field	Description
11–15 CFS	Channel Function Select. This field defines the function to be performed by the channel (see Functions and threads). The Function assigned to the channel has to be compatible with the channel condition encoding scheme, selected by field ETCS.
16 ODIS	Output Disable. This bit enables the channel to have its output forced to the value opposite to OPOL when the output disable input signal corresponding to the channel group that it belongs is active (see eTPU channel output disable signals and Figure 37-305). 0 Turns off the output disable feature for the channel. 1 Turns on the output disable feature for the channel.
17 OPOL	Output Polarity. Determines the output signal polarity. The activation of the output disable signal forces, when enabled by the ODIS bit, the channel output signal to the opposite of this polarity (see Figure 37-305). 0 Output active low (output disable drives output to high). 1 output active high (output disable drives output to low).
18–20 Reserved	Reserved. This field is reserved. This read-only field is reserved and always has the value 0.
21–31 CPBA	Channel n Parameter Base Address. The value of this field times 8 specifies the SDM parameter base host (byte) address for channel n (2-parameter granularity; see SDM organization). As seen by the Host, the channel parameter base (byte) address is (without parameter sign extension) $eTPU\_Base + 0x8000 + CPBA * 8$ or (with parameter sign extension), $eTPU\_Base + 0xC000 + CPBA * 8$.

37.13.5 eTPU Channel n Status Control Register for Engine 2 (eTPU_CCCR_CnSCR_ENG2)

This register gathers the interrupt status bits of the channel and the Function Mode definition (read-write). Bits CIS, CIOS and DTRS for each channel can be also accessed from [eTPU Channel Interrupt Status Register for Engine 2 \(eTPU_GCR_CISR_ENG2\)](#), [eTPU Channel Interrupt Overflow Status Register for Engine 2 \(eTPU_GCR_CIOSR_ENG2\)](#) and [eTPU Channel Data Transfer Request Status Register for Engine 2 \(eTPU_GCR_CDTRSR_ENG2\)](#). Host must write 1 to clear a status bit.

Address: 0h base + 804h offset + (16d × i), where i=0d to 31d

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	CIS	CLOS	0						DTRS	DTROS	0					
W	w1c	w1c							w1c	w1c						
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	IPS	OPS	0	0												FM
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

eTPU_CCCR_CnSCR_ENG2 field descriptions

Field	Description
0 CIS	Channel Interrupt Status. These bits are mirrored in eTPU Channel Interrupt Status Register for Engine 2 (eTPU_GCR_CISR_ENG2) . Also, see Channel Interrupt and Data Transfer Requests . When read: 0.....Channel has no pending interrupt to the Host CPU. 1.....Channel has a pending interrupt to the Host CPU. When written: 0.....Keep interrupt status bit unaltered.

Table continues on the next page...

eTPU_CCCR_CnSCR_ENG2 field descriptions (continued)

Field	Description
	1.....Clear interrupt status bit.
1 CIOS	Channel Interrupt Overflow Status. These bits are mirrored in eTPU Channel Interrupt Overflow Status Register for Engine 2 (eTPU_GCR_CIOSR_ENG2). Also, see Interrupt and Data Transfer Request overflow. When read: 0.....Interrupt overflow negated for this channel. 1.....Interrupt overflow asserted for this channel. When written: 0.....Keep status bit unaltered. 1.....Clear status bit.
2–7 Reserved	Reserved. This field is reserved. This read-only field is reserved and always has the value 0.
8 DTRS	Data Transfer Request Status. These bits are mirrored in eTPU Channel Interrupt Status Register for Engine 2 (eTPU_GCR_CISR_ENG2). Also, see Channel Interrupt and Data Transfer Requests. When read: 0.....Channel has no pending data transfer request. 1.....Channel has a pending data transfer request. When written: 0.....Keep status bit unaltered. 1.....Clear status bit.
9 DTROS	Data Transfer Request Overflow Status. These bits are mirrored in eTPU Channel Data Transfer Request Overflow Status Register for Engine 2 (eTPU_GCR_CDTROSR_ENG2). Also, see Channel Interrupt and Data Transfer Requests. When read: 0.....Data transfer request overflow negated for this channel. 1.....Data transfer request overflow asserted for this channel. When written: 0.....Keep status bit unaltered. 1.....Clear status bit.
10–15 Reserved	Reserved. This field is reserved. This read-only field is reserved and always has the value 0.
16 IPS	Channel Input Pin State. This bit shows the current value of the filtered channel input signal state.
17 OPS	Channel Output Pin State.

Table continues on the next page...

eTPU_CCCR_CnSCR_ENG2 field descriptions (continued)

Field	Description
	This bit shows the current value driven in the channel output signal, including the effect of the external output disable feature (see eTPU channel output disable signals). If the channel input and output signals are connected to the same pad, OPS reflects the value driven to the pad. This is not necessarily the actual pad value, which drives the value in the bit IPS.
18 Reserved	Reserved. This field is reserved. This read-only field is reserved and always has the value 0.
19–29 Reserved	Reserved. This field is reserved. This read-only field is reserved and always has the value 0.
30–31 FM	Channel Function Mode. Each function uses this field for a specific configuration. These bits can be tested by microengine code (see Conditional/unconditional branch). These bits are equivalent to the TPU/TPU2/TPU3 Host Sequence (HSQ) bits.

37.13.6 eTPU Channel n Host Service Request Register for Engine 2 (eTPU_CCCR_CnHSRR_ENG2)

This register is used by the Host to issue service requests to the channel.

Address: 0h base + 808h offset + (16d × i), where i=0d to 31d

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0																												HSR			
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	

eTPU_CCCR_CnHSRR_ENG2 field descriptions

Field	Description
0–28 Reserved	Reserved. This field is reserved. This read-only field is reserved and always has the value 0.
29–31 HSR	Host Service Request This field is used by the Host CPU to request service to the channel (see Host Service Requests). HSR value turns to 000 automatically at the end of microengine service for that channel, but only if the thread started due to an HSR. Host should write HSR greater than 0 only when HSR=0. Writing HSR=000 withdraws a pending request if scheduler did not begin to resolve the Entry Point yet, but it does not abort the service thread from that point on. For more details, see Entry Points and Host Service Requests . 000 No Host Service Request pending.

Table continues on the next page...

eTPU_CCCR_CnHSRR_ENG2 field descriptions (continued)

Field	Description
001	Function-dependent Host Service Request pending.
---	---
111	Function-dependent Host Service Request pending.

37.14 Shared Subsystem Integration memory map and registers

Shared Subsystems Integration Registers configure aspects of how the shared modules interface with the eTPU. There is one integration register for each subsystem unit, which can be, depending on the specific subsystem, a whole IP module, a sub-module, or a channel within a module. For more details, see [Shared Subsystems support](#).

eTPU_SSIR memory map

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/page
C00	eTPU Shared Subsystem n Integration Register (eTPU_SSIR_ETPUSSSIR0)	32	R/W	0000_0000h	37.14.1/1598
C04	eTPU Shared Subsystem n Integration Register (eTPU_SSIR_ETPUSSSIR1)	32	R/W	0000_0000h	37.14.1/1598
C08	eTPU Shared Subsystem n Integration Register (eTPU_SSIR_ETPUSSSIR2)	32	R/W	0000_0000h	37.14.1/1598
C0C	eTPU Shared Subsystem n Integration Register (eTPU_SSIR_ETPUSSSIR3)	32	R/W	0000_0000h	37.14.1/1598
C10	eTPU Shared Subsystem n Integration Register (eTPU_SSIR_ETPUSSSIR4)	32	R/W	0000_0000h	37.14.1/1598
C14	eTPU Shared Subsystem n Integration Register (eTPU_SSIR_ETPUSSSIR5)	32	R/W	0000_0000h	37.14.1/1598
C18	eTPU Shared Subsystem n Integration Register (eTPU_SSIR_ETPUSSSIR6)	32	R/W	0000_0000h	37.14.1/1598
C1C	eTPU Shared Subsystem n Integration Register (eTPU_SSIR_ETPUSSSIR7)	32	R/W	0000_0000h	37.14.1/1598
C20	eTPU Shared Subsystem n Integration Register (eTPU_SSIR_ETPUSSSIR8)	32	R/W	0000_0000h	37.14.1/1598
C24	eTPU Shared Subsystem n Integration Register (eTPU_SSIR_ETPUSSSIR9)	32	R/W	0000_0000h	37.14.1/1598

37.14.1 eTPU Shared Subsystem n Integration Register (eTPU_SSIR_ETPUSSSIRn)

This register is used to configure the interface of a subsystem unit with the eTPU.

Address: 0h base + C00h offset + (4d × i), where i=0d to 9d

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0								LENG		0	LCHAN				
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

eTPU_SSIR_ETPUSSSIRn field descriptions

Field	Description
0–23 Reserved	Reserved. This field is reserved. This read-only field is reserved and always has the value 0.
24–25 LENG	Link Engine. LENG specifies the engine that receives a Link Service Request when an interrupt is issued by the Shared Unit, or disables Shared Unit linking. The channel that receives the Link Service Request is specified in the LCHAN field. 00 Shared Unit Link Service Request disabled 01 Engine 1 receives Shared Unit Link Service Request 10 Engine 2 receives Shared Unit Link Service Request 11 Reserved
26 Reserved	Reserved. This field is reserved. This read-only field is reserved and always has the value 0.
27–31 LCHAN	Link Channel. LCHAN specifies the channel linked when an interrupt is issued by the Shared Unit, if enabled in the LENG field. The field is meaningless if LENG disables Shared Unit linking. For more information on Link Service Requests, see Channel Link .

37.15 Functional description

37.15.1 Functions and threads

eTPU processing is event-driven, in the sense that eTPU microcode only runs to service a request from an event. Service Requests may result from the occurrence of any of the following events:

- Host CPU writing a non-zero value to the channel HSR (Host Service Request) field in [eTPU Channel n Host Service Request Register for Engine 1 \(eTPU_CCCR_CnHSRR_ENG1\)](#) and [eTPU Channel n Host Service Request Register for Engine 2 \(eTPU_CCCR_CnHSRR_ENG2\)](#).
- Occurrence of a time base match, an input signal transition, or a specific combination of them (depending on the Channel Mode currently configured).
- A Link Service Request.

A given event is always associated to only one Channel:

- There is one HSR register field for each Channel
- Each signal is associated with only one Channel, which has its own Match registers and independent mode configuration.
- Each Link Service Request can have only one Channel as a target.

Service Request processing is done by a set of microengine routines. A set of related routines that implement a specific channel application is called a **Function**. One or more Functions reside on SCM, limited only by the SCM space available, size of microcode Functions and the number of entry points available. Each Engine can be controlled by up to 32 Functions at a time.

A Function can be assigned to several channels, but only one Function can be assigned to a given Channel at a time. This is defined by the Host through the Channel Configuration Registers (see [Channel Configuration Control memory map and registers](#)).

The term **Thread** will be used hereafter to refer to a service routine of a Function, or its execution. A Thread is constructed of a specific number of microinstructions, typically the code necessary to calculate the next phase of waveform to be input to, or output from, a given channel. Once a Thread begins, its execution cannot be interrupted. A Thread normally finishes when an END microinstruction is executed.

A given Thread is selected and called by the Scheduler depending on the following:

- the type of event that generated the service request.

- the Function assigned to the target channel.
- target channel pin state.
- the state of the channel logic.
- the priority assigned to the target channel, relative to the priorities of other channels with pending service requests

The mechanism to select a thread based on the channel Function and type of event is described in the [Entry Points](#).

The priority mechanism that determines the order of Thread execution amongst pending service requests is described in [Scheduler](#).

37.15.1.1 Entry Points

37.15.1.1.1 Entry Table

Each Thread has its **Entry Point**, which contains the SCM address of its first instruction, besides other information. For a complete Entry Point description, see [Entry Point format](#).

Once the Scheduler chooses a channel among pending Service Requests, the Entry Point is taken from an **Entry Table**, based on the Function assigned for the channel and other conditions. Entry Table layout is shown in the following figure.

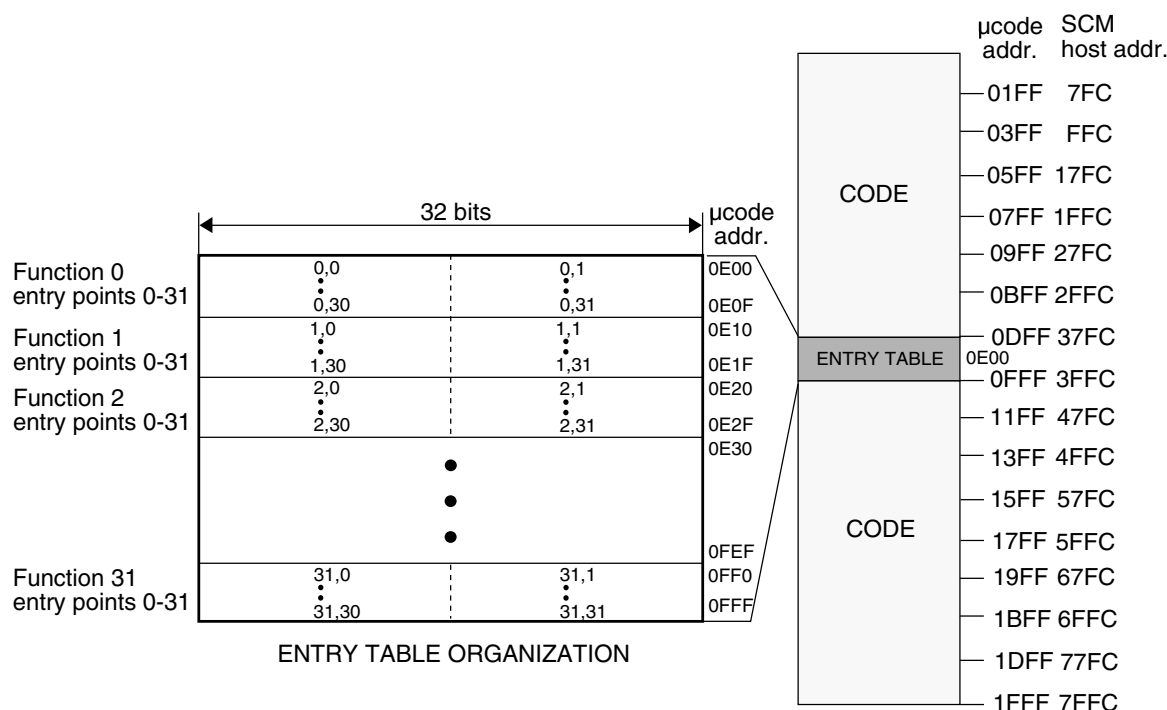


Figure 37-291. Entry Table

The Entry Table is organized by Functions. Each Function can have up to 32 Entry Points of 16 bits each, corresponding to 32 possible Threads per Function. Each Entry Point location in the table corresponds to a combination of events and channel states (see [Entry Point address generation](#)). A single Thread can be associated to more than one combination, having its Entry Point repeated in the table. Each 32-bit word in the Entry Table holds two Entry Points.

Note that the Entry Table can be placed in any SCM address multiple of the Entry Table size, determined by the ETB field in [eTPU Engine 1 Configuration Register \(eTPU_SCR_ECR_ENG1\)](#) and [eTPU Engine 2 Configuration Register \(eTPU_SCR_ECR_ENG2\)](#). However, it is recommended to place the Entry Table at the start of the SCM to get continuous code memory and to ease the eventual migration of the code from larger parts down to smaller ones without rearranging the binary image, but this is not a restriction. Unused Entry Points may be used for microcode, so this organization extends the microcode continuous area to the unused area of the Entry Table. For this purpose, Function numbers should be selected from 0 up to 31. If, for example, only 8 functions are implemented, only the Entry Table locations for Functions 0 to 7 are used, and the Entry Table locations for functions 8 to 31 can be used as microinstruction memory (adding extra continuous 1536 bytes for microprogram usage).

One way of implementing different sets of Functions is having more than one Entry Table, and configuring the eTPU with the appropriate one for the application by changing the ETB field in [eTPU Engine 1 Configuration Register \(eTPU_SCR_ECR_ENG1\)](#) and [eTPU Engine 2 Configuration Register \(eTPU_SCR_ECR_ENG2\)](#). Note that the Engines can use different Entry Tables, with or without the same set of Functions.

37.15.1.1.2 Entry Point address generation

The Entry Point address within the Entry Table is determined by the Function assigned to the Channel, the state of the Channel, the type of event, and the condition encoding scheme. Together with the Entry Table base address, they form the Entry Point Address at the SCM, as shown in the following figure.

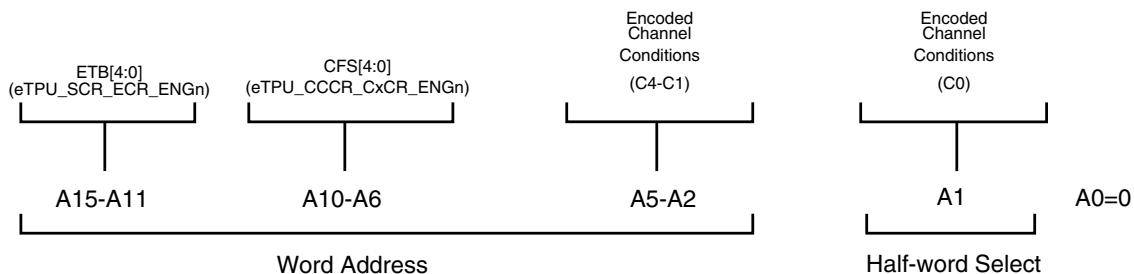


Figure 37-292. Entry Point Address (host address offset)

The type of event and channel state are coded in the Encoded Channel Conditions field C4-C0, according to one of two encoding schemes:

- Standard Entry Table Condition encoding scheme, shown in [Table 37-333](#), which privileges Host Service Requests.
- Alternate Entry Table Condition encoding scheme, shown in [Table 37-334](#), which focus on other events and state decoding.

The events that take part on condition encoding generate a Service Request, and have four origins:

1. Match Recognition (caused by greater/equal match, or equal-only, between the value TCR1/2 and the value stored in the channel match registers). eTPU channels support single and double match in various modes of match recognition; see [Match Recognition](#).
2. Transition Detect Service Request (channel input signal transition detection of a selected edge). The eTPU channels support single and double transition, which together with the double match options provide various modes of transition detection; see [Transition detection and Time Base Capture](#).

3. Channel Linking Service Request (microcode writing the channel number to the LINK register). Link service request allows one channel to activate another (see [Channel Link](#)). Links may also come from Shared Subsystem units (see [Shared Subsystems support](#)).
4. Host Service Request (Host writes a non-zero value to the HSR bits of the channel; see [Host Service Requests](#)).

Note

Even if a Transition or Match Service Request is inhibited (by channel mode/state or SRI), the Transition Detection and Match Recognition are taken into account for condition encoding. That is, the MRLA/B and TDLA/B flags are used, not their respective Service Requests.

Columns Host Request Bits, Link Request, MatchA/TransB, and MatchB/TransA determine the type of event. A non-zero value in these columns represents the recognition of the event, while "x" indicates that its recognition is irrelevant. Values 1 and 0 mean that event was recognized or not, respectively. Note that Match and Transition events may occur and not be recognized, and in this case it assumes value 0 for the condition encoding. The recognition of such an occurred event depends on the channel mode assigned and other conditions, as described in [Enhanced Channels](#).

The **Host Service Request Bits** column refers to the value written by the Host CPU to the Host Service Request Registers ([eTPU Channel n Host Service Request Register for Engine 1 \(eTPU_CCCR_CnHSRR_ENG1\)](#) and [eTPU Channel n Host Service Request Register for Engine 2 \(eTPU_CCCR_CnHSRR_ENG2\)](#)) of the Channel being serviced. Note that the bits on this row are coded (3-bit representation). If the value of HSR is not zero, then the Host actually requested service.

The **Link Request** column refers to the occurrence of a Channel Link or SSS Link requests.

The **MatchA/TransB** column refers to the recognition of either a Match event specified by MatchA channel register or the detection of a channel input signal event specified by the IPACB configuration register (see [Pin Control Registers](#)).

The **MatchB/TransA** column refers to the recognition of either a Match event specified by MatchB channel register or the detection of a channel input signal event specified by the IPACA configuration register (see [Pin Control Registers](#)).

For the channel input signal, MatchA and MatchB provide double timeout conditions which depend on the channel mode programming (see [Channel Modes](#)). If the channel is used for output only, there are no transition detections, so the MatchB/TransA column

represents only Match B, and MatchA/TransB column the Match A. In this case Match A and Match B are separated to give better state resolution in double match output functions. For more information about channel requests refer to [Enhanced Channels](#).

Besides those events, the following channel state conditions help to determine the Entry Point:

1. Channel Flags 0 and 1: these are channel-internal flags (not in SDM) associated with a channel. Their values are set by microcode (see [Channel flags operations](#)).
2. Input Pin state or Output Flip Flop: the state (0 or 1) of the channel input signal after the Enhanced Filter (see [Enhanced Digital Filter - EDF](#)), or the state driven to the output signal. Which one (input or output) is used is selected by the ETPD field in [eTPU Channel n Configuration Register for Engine 1 \(eTPU_CCCR_CnCR_ENG1\)](#) and [eTPU Channel n Configuration Register for Engine 2 \(eTPU_CCCR_CnCR_ENG2\)](#).

The two Entry Table Condition encoding schemes combine events and state conditions differently, as detailed in the following sections.

37.15.1.1.3 Standard condition encoding scheme

In this scheme, shown in the following table, all 7 HSR combinations are used and other event type columns are marked "x" when HSR is non-zero, indicating that Host Service Request has priority over any other type of event. However, when an HSR service thread is called (entry numbers 0 to 9), other events may also have been recognized, and it is microcode responsibility to check them.

When HSR is 0, i.e., Host did not issue a Service Request to the channel, the other event conditions, the input signal state and channel flags determine the Entry Point. Note that channel flag 1 does not influence the encoding in this scheme.

Table 37-333. Standard channel condition encoding scheme

No.	Encoded channel conditions [C4-C0]	Host Service Request bits	Link request	MatchA/ TransB	MatchB/ TransA	In/output pin state ¹	Channel Flag1	Channel Flag0
0	001	00000	x	x	x	0	x	0
1	00001	001	x	x	x	0	x	1
2	00010	001	x	x	x	1	x	0
3	00011	001	x	x	x	1	x	1
4	00100	010	x	x	x	x	x	x
5	00101	011	x	x	x	x	x	x
6	00110	100	x	x	x	x	x	x

Table continues on the next page...

Table 37-333. Standard channel condition encoding scheme (continued)

No.	Encoded channel conditions [C4-C0]	Host Service Request bits	Link request	MatchA/TransB	MatchB/TransA	In/output pin state ¹	Channel Flag1	Channel Flag0
7	00111	101	x	x	x	x	x	x
8	01000	110	x	x	x	x	x	x
9	01001	111	x	x	x	x	x	x
10	01010	000	1	1	1	x	x	0
11	01011	000	1	1	1	x	x	1
12	01100	000	0	0	1	0	x	0
13	01101	000	0	0	1	0	x	1
14	01110	000	0	0	1	1	x	0
15	01111	000	0	0	1	1	x	1
16	10000	000	0	1	0	0	x	0
17	10001	000	0	1	0	0	x	1
18	10010	000	0	1	0	1	x	0
19	10011	000	0	1	0	1	x	1
20	10100	000	0	1	1	0	x	0
21	10101	000	0	1	1	0	x	1
22	10110	000	0	1	1	1	x	0
23	10111	000	0	1	1	1	x	1
24	11000	000	1	0	0	0	x	0
25	11001	000	1	0	0	0	x	1
26	11010	000	1	0	0	1	x	0
27	11011	000	1	0	0	1	x	1
28	11100	000	1	0	1	x	x	0
29	11101	000	1	0	1	x	x	1
30	11110	000	1	1	0	x	x	0
31	11111	000	1	1	0	x	x	1
		Host Service Request						

1. The ETPD field in [eTPU Channel n Configuration Register for Engine 1 \(eTPU_CCCR_CnCR_ENG1\)](#) and [eTPU Channel n Configuration Register for Engine 2 \(eTPU_CCCR_CnCR_ENG2\)](#) selects between input and output pin state.

37.15.1.1.4 Alternate condition encoding scheme

This scheme is shown in the following table. Because the HSR bits cannot be tested by microcode, only three distinct Host Service Request can be used:

1. HSR=010 or 011, which are coded into the same Entry Points (0 to 3)
2. HSR=100, 101 or 001, which are all coded into Entry Point 4

3. HSR=110 or 111, which are both coded into Entry Point 5

The remaining Entry Points use both channel flags for better state decoding, making this scheme better suited for Functions which need more states and/or faster state decoding, without needing many HSRs.

Table 37-334. Alternate channel condition encoding scheme

No.	Encoded channel conditions [C4-C0]	Host Service Request bits	Link request	MatchA/TransB	MatchB/TransA	In/Output pin state ¹	Channel Flag1	Channel Flag0
0	00000	01x	x	x	x	0	x	0
1	00001	01x	x	x	x	0	x	1
2	00010	01x	x	x	x	1	x	0
3	00011	01x	x	x	x	1	x	1
4	00100	10x/001	x	x	x	x	x	x
5	00101	11x	x	x	x	x	x	x
6	00110	000	1	0	0	0	x	x
7	00111	000	1	0	0	1	x	x
8	01000	000	x	1	0	0	0	0
9	01001	000	x	1	0	0	0	1
10	01010	000	x	1	0	0	1	0
11	01011	000	x	1	0	0	1	1
12	01100	000	x	1	0	1	0	0
13	01101	000	x	1	0	1	0	1
14	01110	000	x	1	0	1	1	0
15	01111	000	x	1	0	1	1	1
16	10000	000	x	0	1	0	0	0
17	10001	000	x	0	1	0	0	1
18	10010	000	x	0	1	0	1	0
19	10011	000	x	0	1	0	1	1
20	10100	000	x	0	1	1	0	0
21	10101	000	x	0	1	1	0	1
22	10110	000	x	0	1	1	1	0
23	10111	000	x	0	1	1	1	1
24	11000	000	x	1	1	0	0	0
25	11001	000	x	1	1	0	0	1
26	11010	000	x	1	1	0	1	0
27	11011	000	x	1	1	0	1	1
28	11100	000	x	1	1	1	0	0
29	11101	000	x	1	1	1	0	1
30	11110	000	x	1	1	1	1	0

Table continues on the next page...

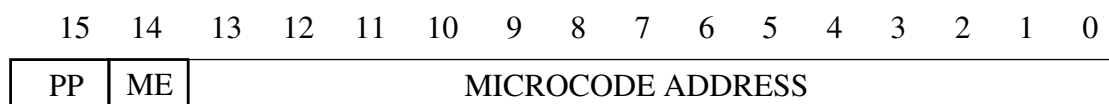
Table 37-334. Alternate channel condition encoding scheme (continued)

No.	Encoded channel conditions [C4-C0]	Host Service Request bits	Link request	MatchA/TransB	MatchB/TransA	In/Output pin state ¹	Channel Flag1	Channel Flag0
31	11111	000	x	1	1	1	1	1
		Host Service Request						

1. The ETPD field in [eTPU Channel n Configuration Register for Engine 1 \(eTPU_CCCR_CnCR_ENG1\)](#) and [eTPU Channel n Configuration Register for Engine 2 \(eTPU_CCCR_CnCR_ENG2\)](#) selects between input and output pin state.

37.15.1.1.5 Entry Point format

Entry Point information includes a Preload-Parameter selection field, a Match Enable field, and the first microcode address of the thread. The Entry Point format is illustrated in the following figure.

**Figure 37-293. Entry Point format**

PP - Preload Parameter

PP indicates which pair of channel parameters are loaded into registers P and DIOB from the SDM prior to the execution of a thread. Preloading occurs during the time-slot transition period (see [Time Slot Transition](#)).

- 1 = microengine register P is preloaded from parameter 2 and DIOB from parameter 3.
- 0 = microengine register P is preloaded from parameter 0 and DIOB from parameter 1.

The parameter numbers are offsets from the channel parameter base address. For more info, see [Parameter Access](#).

ME - Match Enable

ME specifies whether match event recognitions are enabled or disabled for the thread associated with the entry point during the thread execution. If they are disabled, a match recognition can only occur after channel service. For more details refer to [Match Recognition](#).

- 1 = Matches are enabled during the thread.
- 0 = Matches are disabled during the thread.

The disabling of Match A/B recognition by MEF is dependent on IPACA/B configuration on the serviced channel (see [Pin Control Registers](#)). If IPACA=1xx, Match A is not disabled by ME=0. Likewise, IPACB=1xx overrides the effect of ME on Match B to "always on" If IPACA/B=0xx, Match A/B is disabled for two microcycles during TST (see [Time Slot Transition](#)) and is re-enabled when Entry Point is loaded, if ME=1. Note that if the comparator is in equal-only mode and the time base reaches the value of the Match register during the time that recognition is disabled (beginning of TST, plus whole thread if ME=0), the match recognition is lost. If the comparator is in greater-equal mode, the match event may be recognized after the disabling period if it satisfies the "greater-than" condition.

MICROCODE ADDRESS - Starting Microcode Address

This field specifies the microcode address on which the thread is to begin execution.

37.15.1.2 Time Slot Transition

The Time Slot Transition period (also called TST for short) is the interval between the servicing of two channels, during which all Channel-specific context is loaded for the new serviced Channel. The primary tasks completed during this period include:

- Set MEF for the TST wait-states, if any.
- Reset MEF for the first two microcycles (after eventual wait-states).
- Update of the CHAN register with the number of the new channel to be serviced.
- Parallel update of ERTA and ERTB from CaptureA and CaptureB registers of the new serviced channel.
- Sampling of the branch conditions of the new channel to be serviced into the branch logic (this means flags TDLA/B, MRLA/B, LSR, FM[1], FM[0], and PSS). The branch conditions are coherent with the timebase capture values sampled into ERTA/B (if MRLA/B, TDLA/B are set at the same time of the sampling, either both old flag state and capture values are sampled, or both new values are sampled).
- Formation of the entry point address.
- Copy the ME bit in the Entry Point into MEF.
- Access to the entry point location and getting the first microinstruction address.
- Preload of two parameters from the SDM into P (32 bits) and DIOB (24 bits).
- Fetch the first instruction of the thread to be executed for the new channel.
- Preset the RAR register value (see [RAR Register](#)).

The preload operation is 32-bit wide for P and 24-bit wide for DIOB. The P register is loaded with all the 32-bit parameter. The DIOB register is loaded with the lower 24-bits of the parameter. The microcode can switch at any time to access the lower 24-bits, upper byte, or all the 32-bits of any parameter in the SDM. Preload of P-DIOB pair of parameters is atomic with respect to Host and CDC accesses, and so are coherent with their dual-parameter coherent transfers. For more details see [Parameter sharing and coherency](#).

No instructions are executed at the Engine where the time slot transition period occurs, but the other Engine can execute normally. Match A/B is unconditionally disabled on the first two TST microcycles, if IPACA/B=0xx (respectively). During the rest of time slot transition, match recognition can be disabled or not, depending on IPACA/B field and ME. See [Match Recognition](#).

Time Slot Transition takes a minimum of 3 microcycles (6 eTPU clocks), which may be extended due to SDM arbitration wait-states for the first preload access (see [SDM arbitration](#)). When no wait-states are received ([Figure 37-294](#)), DIOB is preloaded twice, one for each PP value, and the correct value remains in DIOB when the Entry Point is loaded. [Figure 37-295](#) and [Figure 37-296](#) show the timing for one and two wait-states, respectively.

Registers B, C, D and SR are not altered by TST and keep their values from the previous thread. The values of registers A, MACL and MACH are not guaranteed at the thread start.

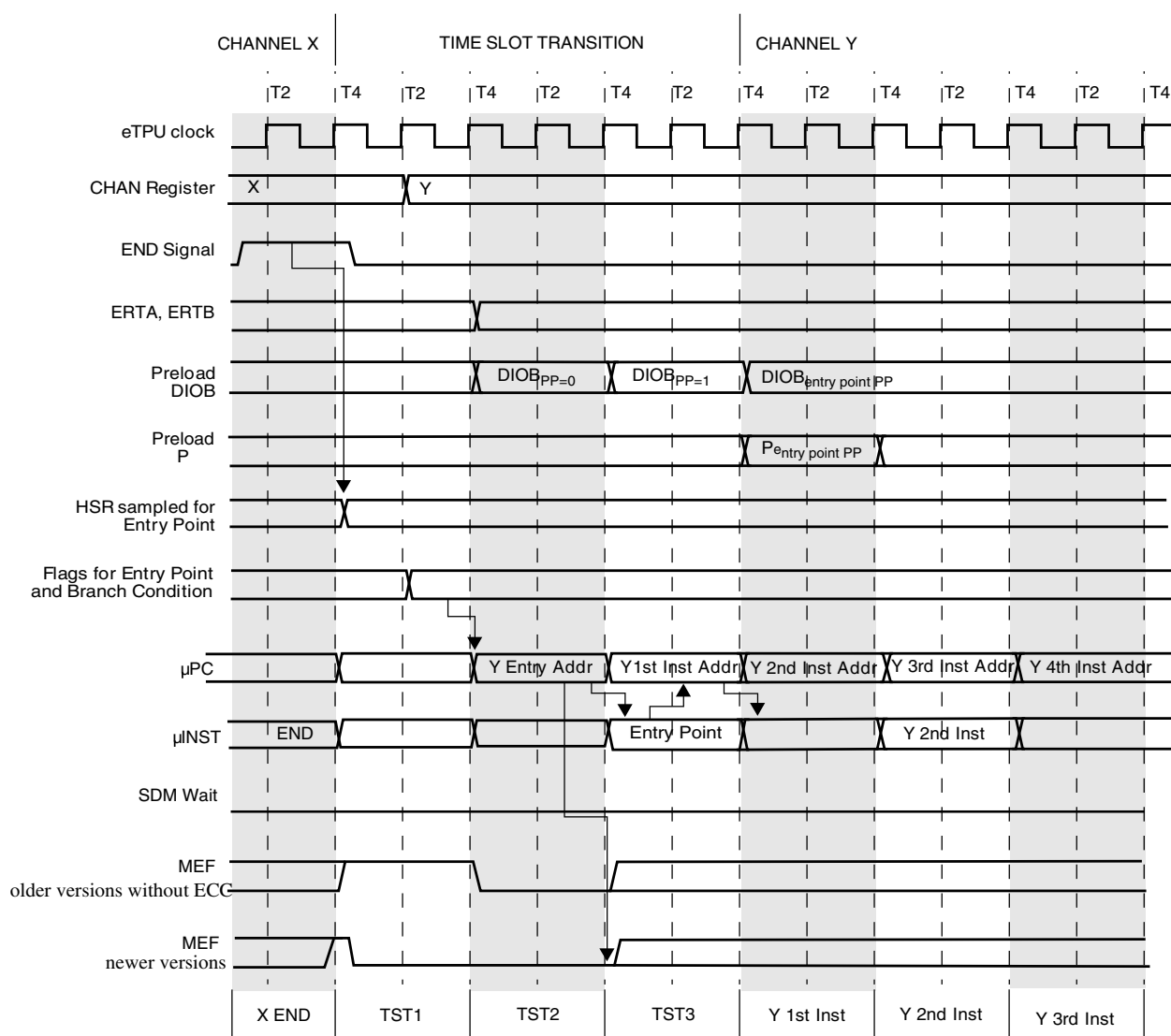


Figure 37-294. TST Timing - No Wait-states

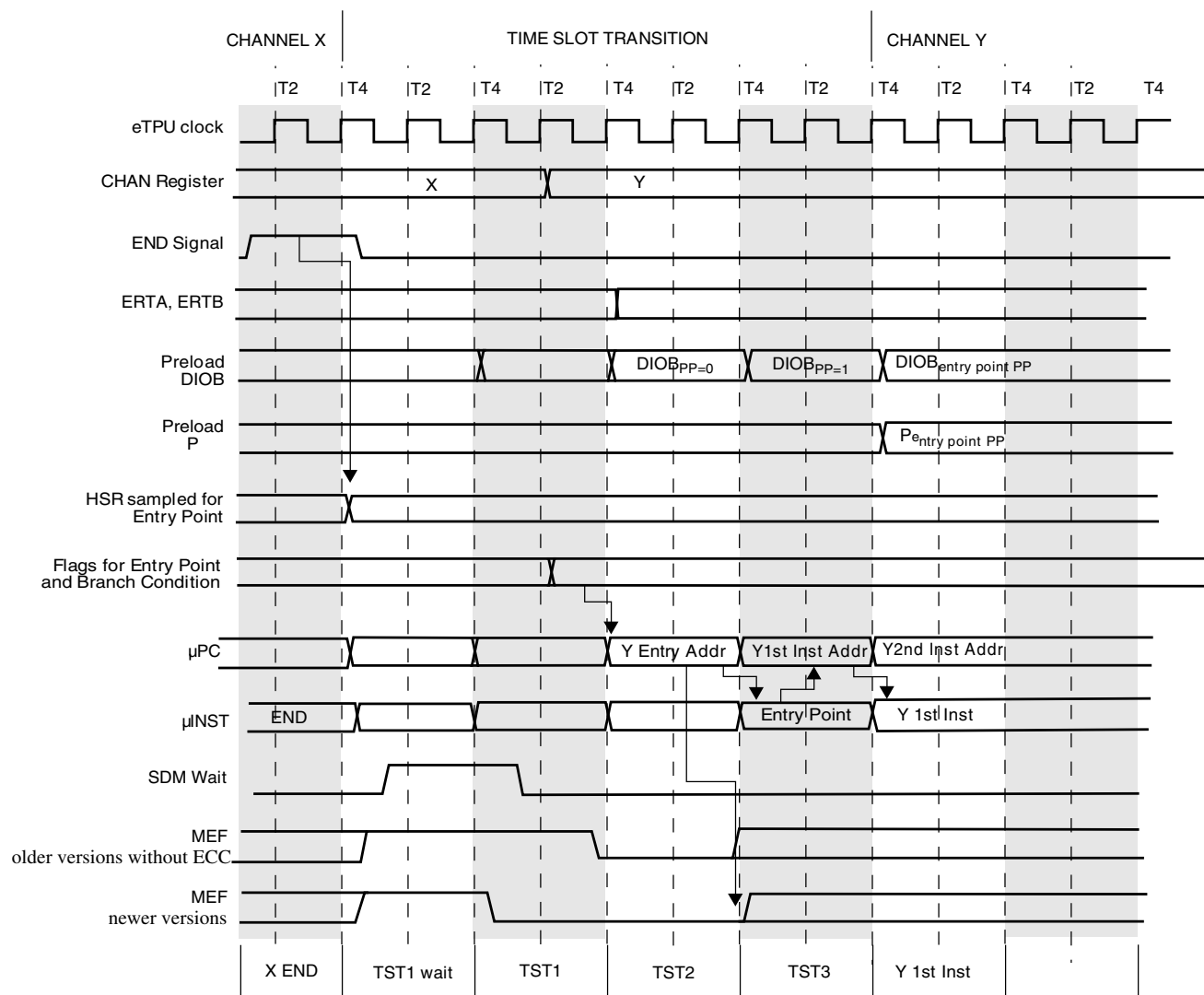


Figure 37-295. TST Timing - 1 Wait-State

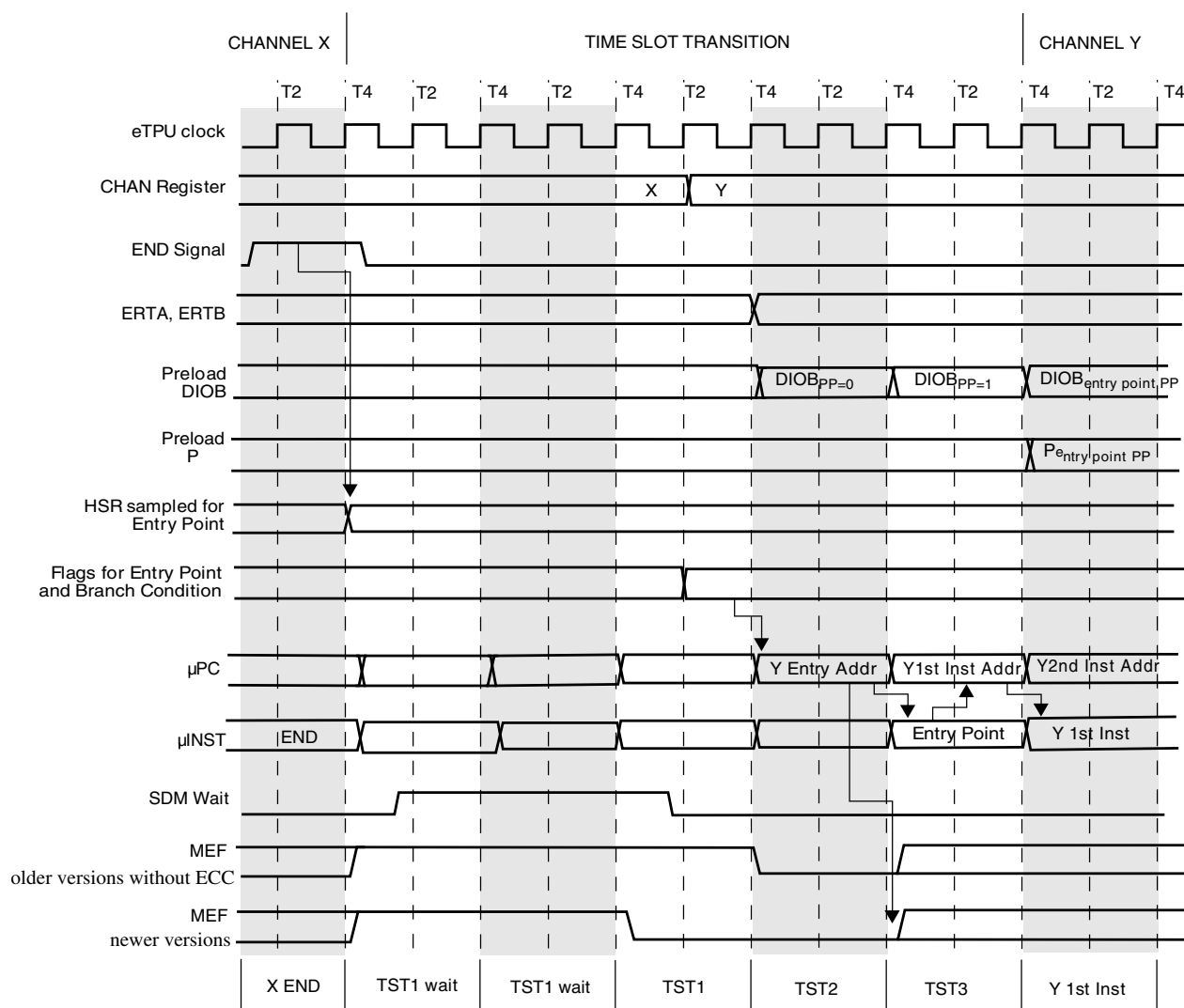


Figure 37-296. TST Timing - 2 Wait-states

For more information on Channel-specific registers and flags, refer to [Enhanced Channels](#). For more information on P, ERTA/B and DIOB registers refer to [Registers](#).

37.15.1.2.1 Thread gap

The MEF timing has changed from previous eTPU/eTPU2 versions, now being reset for two microcycles from the start of TST.¹² Therefore, two consecutive threads with ME=0 serving the same channel could block any match until the end of the second thread (see [Figure 37-294](#)). This could cause latency problems and also break compatibility with legacy code, so the thread transition logic was modified to introduce a one-microcycle gap between consecutive threads servicing the same channel, as shown in the following

12. the new MEF timing is found on all eTPU2 instances starting from the first version with Memory Error Support; all previous eTPU2 versions maintain the former MEF timing (see [Legacy Time Slot Transition Timing](#)).

figure. During the gap MEF is set to one, regardless of the previous or the next entry point ME field. The thread gap is inserted only between threads serving the same channel.

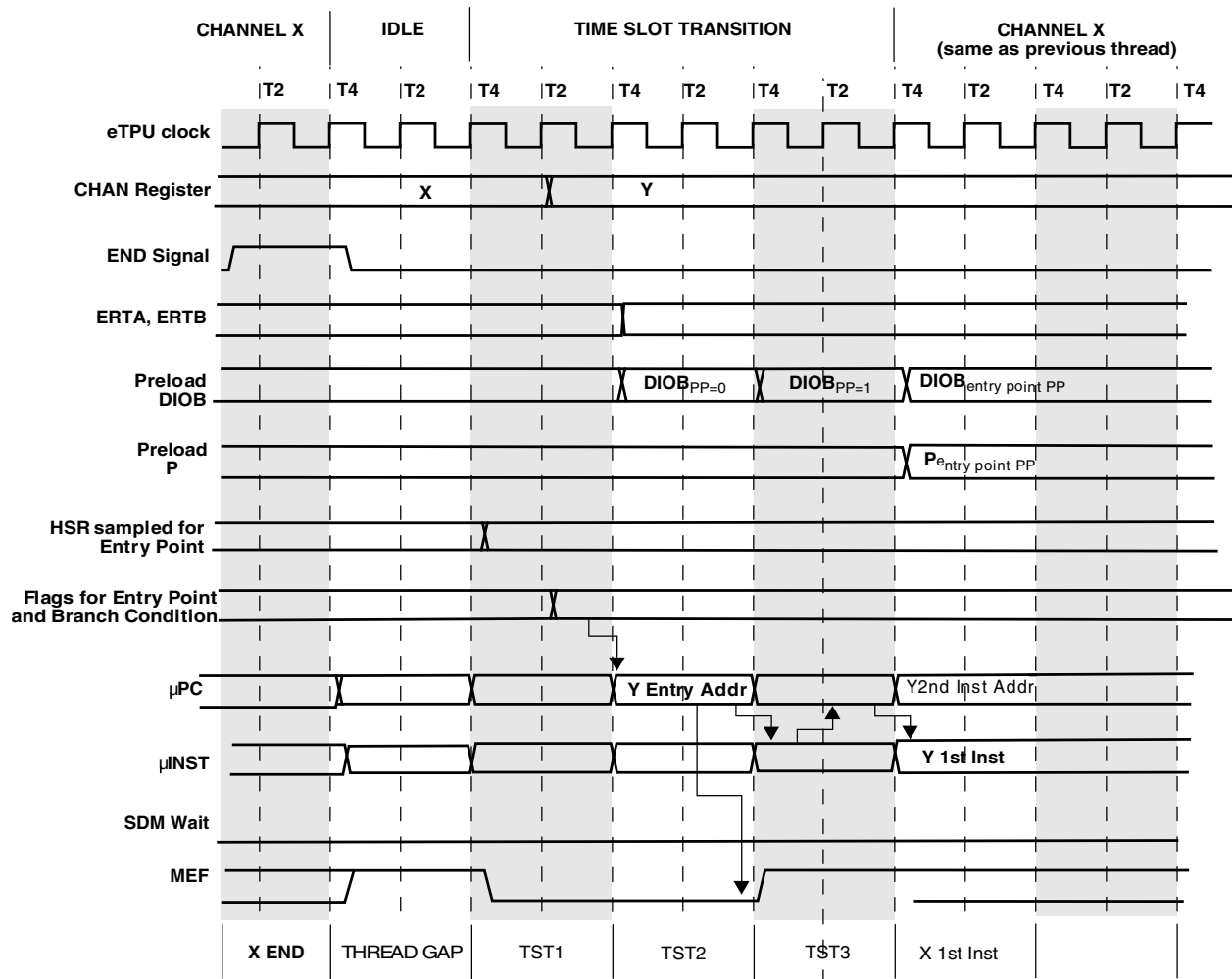


Figure 37-297. Thread gap

37.15.1.3 Thread ending

Threads can finish by either:

- an instruction with the END field active (see [Ending Current Thread - END](#)).
- a forced END by host writing to the FEND field in [eTPU Engine 1 Configuration Register \(eTPU_SCR_ECR_ENG1\)](#) and [eTPU Engine 2 Configuration Register \(eTPU_SCR_ECR_ENG2\)](#).

- a forced END caused by Watchdog timeout (see [Watchdog](#)).
- a forced END caused by a non-correctable memory error (see [Memory error support](#)).

37.15.1.4 Watchdog

Each engine has a watchdog mechanism to prevent a thread or a sequence of threads from running too long, impacting the latency of the other channel services. The watchdog is configured through [eTPU Watchdog Timer Register for Engine 1](#) (`eTPU_ER_WDTR_ENG1`) and [eTPU Watchdog Timer Register for Engine 2](#) (`eTPU_ER_WDTR_ENG2`). When the watchdog is enabled, an internal counter increments on each microcycle when a thread is executing. If the count is greater than the value specified in the WDCNT field in [eTPU Watchdog Timer Register for Engine 1](#) (`eTPU_ER_WDTR_ENG1`) and [eTPU Watchdog Timer Register for Engine 2](#) (`eTPU_ER_WDTR_ENG2`) and a thread is still executing, the watchdog:

1. If the WMO field in [eTPU Watchdog Timer Register for Engine 1](#) (`eTPU_ER_WDTR_ENG1`) and [eTPU Watchdog Timer Register for Engine 2](#) (`eTPU_ER_WDTR_ENG2`) is negated, forces an END of the thread, allowing other channels to be serviced
2. Sets the WDS status bit of the serviced channel in [eTPU Watchdog Status Register for Engine 1](#) (`eTPU_GCR_WDSR_ENG1`) and [eTPU Watchdog Status Register for Engine 2](#) (`eTPU_GCR_WDSR_ENG2`).
3. If the WMO field in [eTPU Watchdog Timer Register for Engine 1](#) (`eTPU_ER_WDTR_ENG1`) and [eTPU Watchdog Timer Register for Engine 2](#) (`eTPU_ER_WDTR_ENG2`) is negated, the channel is disabled, not initiating any thread until its WDS bit is cleared (however, its CPR field in [eTPU Channel n Configuration Register for Engine 1](#) (`eTPU_CCCR_CnCR_ENG1`) and [eTPU Channel n Configuration Register for Engine 2](#) (`eTPU_CCCR_CnCR_ENG2`) is not changed).
4. Issues a Global Exception and sets [WDTO1](#) (engine 1) or [WDTO2](#) (engine 2), regardless of the WMO bit state.

If the WMO field in [eTPU Watchdog Timer Register for Engine 1](#) (`eTPU_ER_WDTR_ENG1`) and [eTPU Watchdog Timer Register for Engine 2](#) (`eTPU_ER_WDTR_ENG2`) is asserted, the watchdog can be used to monitor the execution time threshold of a thread or thread sequence without interfering with the eTPU code execution.

The watchdog can be configured in one of the following modes, defining how the internal watchdog count is reset:

- **Thread Length Mode:** the watchdog count is reset at the end of each thread.
- **Busy Length Mode:** the watchdog count is reset when the microengine goes idle. A sequence of threads, one right after another, keeps the count running. The counter is also reinitialized when a thread is forced to end, so that a new count begins if another TST initiates at the following microcycle.

The following applies to the watchdog mechanism:

- microcycles during TST and SDM access wait-states (on TST or instruction execution) are counted.
- if the watchdog count equals WDCNT in the last microinstruction (with SDM wait-states or not) of a thread servicing a channel, its WDS bit is not set.
- if the watchdog count expires (gets greater than WDCNT) during the TST, the thread is forced end on its first instruction.
- the watchdog count does not wrap, so that a thread (in thread length mode) or a thread sequence (in busy length mode) that lasts for more than the maximum value of WDCNT does get a forced end.

Note

Watchdog must not be enabled when the microengine enters halt mode.

The counter does not run when the engine is stopped, and resets when the watchdog is disabled.

37.15.2 Host interface

37.15.2.1 System configuration

System Configuration Registers are described in [System Configuration memory map and registers](#). Detailed explanation of the configured functionalities is found throughout [Functional description](#) and a specification for the initial configuration sequence is found in [Configuration sequence](#).

37.15.2.2 Interrupts and data transfer requests

37.15.2.2.1 Interrupt types and sources

Each one of the eTPU channels can be a source of two requests: **Channel Interrupt** request and **Data Transfer Request**. Channel Interrupts are targeted to a Host CPU. Data Transfer Requests may be targeted to a data transfer module (for example, a DMA controller). Interrupt and Data Transfer registers are used by the Host to enable interrupts and data transfer requests, indicate their status and service them. Interrupt and Data Transfer requests have the same sets of registers and external signals, and are handled in the same way. They differ only by the fact that Data Transfer Requests are also cleared by the assertion of respective DMA completion acknowledge line.

Note

Interrupt and Data Transfer requests can be cleared even when Engines are in Module Disable Mode, through the Global Channel Registers, and also DMA completion for Data Transfer requests.

Channel Interrupts and Data Transfer Requests can only be issued by eTPU microcode, through one of the Channel Control instruction fields (see [Channel Interrupt and Data Transfer Requests](#)).

Both Channel Interrupt and Data Transfer requests can be individually enabled for each channel.

eTPU Interrupt and Data Transfer Registers are mirrored in two organizations: grouped by Channel and grouped by type (interrupt status, interrupt enable, data transfer status, data transfer enable). This allows either "channel-oriented" or "bundled channel" Host interrupt service schemes, or a combination of them. For a detailed description, refer to the Channel Register Layout and Global Channel Registers sections.

eTPU can also assert a **Global Exception** interrupt indicating a global illegal state. There are six possible sources for a Global Exception:

- Execution of an illegal instruction by the microengine (see [Illegal instructions](#)). This Global Exception source is flagged by [ILF1](#) and [ILF2](#).
- An SCM signature mismatch detected by the Multiple Input Signature Calculator - MISC. See [SCM Test - Multiple Input Signature Calculator](#). This source is flagged by [SCMMISF](#).

- Microcode request, through microinstruction field CIRC (see [Channel Interrupt and Data Transfer Requests](#)). This Global Exception source is flagged by [MGE1](#) (Engine 1) and [MGE2](#) (Engine 2). The cause of this illegal state is application-dependent. The microcode may write an error code into the SDM to indicate the cause of the exception, for instance.
- A Watchdog timeout, flagged by [WDTO1](#) and [WDTO2](#) (see [Watchdog](#)).
- An SDM or SCM non-correctable error on a microengine or host access, flagged by [SDMERR](#) and [SCMERR](#), respectively (see [Memory error support](#)).
- Bus errors on microengine accesses to the Shared Subsystem Area, flagged by [SSAERR](#) (see [Shared Subsystems support](#)).
- The latency measured by any of either engine's Latency Monitors is greater than the specified maximum, flagged by [LTO1](#) (Engine 1) and [LTO2](#) (Engine 2) (see [Latency Monitor](#)).

Global Exceptions cannot be directly disabled within eTPU, except by disabling its sources (Memory Error, SSS, MISC and microcode), and it is cleared by writing 1 to [GEC](#). Clearing Global Exception clears all Global Exception source status bits (ILF1, ILF2, SCMMISF, MGE1, MGE2, SSAERR, SCMERR, SDMERR). If GEC is written 1 at the same time any of the sources issues a Global Exception, both the interrupt and the status bit of that source remains asserted. The assertion of Global Exception by one of the sources above does not prevent the others from asserting it too, so any number of them, in any combination, can be flagged.

Note

There can be a race between the clear of a Global Exception and occurrence of a new set condition, such that the set happens just before the clear and cannot be sensed by the Host. Therefore, Global Exception cannot be used as a normal interrupt source. It should only be used for emergency procedures.

Memory Error Correction support (see [Memory error support](#)) has a distinct interrupt source, which can activate on any correctable error occurrence, either to SCM or SDM accesses, but does not activate on non-correctable errors (which assert Global Exception instead). Memory Error Correction interrupt assert when it is enabled through [MECIE](#) (see [eTPU Memory Error Status Register \(eTPU_MESR_MESR\)](#)) and either DCERR or CCERR in [eTPU Memory Error Status Register \(eTPU_MESR_MESR\)](#) is asserted. The interrupt is negated when both DCERR and CCERR are negated, or MECIE is negated.

37.15.2.2.2 Interrupt and Data Transfer Request overflow

If a Channel Interrupt was issued, its status bit is still set, and microcode issues another Channel Interrupt, the Interrupt Overflow status bit is set for that channel. Interrupt Overflow status can be checked by the Host in the CIOS field of [eTPU Channel n Status Control Register for Engine 1 \(eTPU_CCCR_CnSCR_ENG1\)](#) and [eTPU Channel n Status Control Register for Engine 2 \(eTPU_CCCR_CnSCR_ENG2\)](#), mirrored in [eTPU Channel Interrupt Overflow Status Register for Engine 1 \(eTPU_GCR_CIOSR_ENG1\)](#) and [eTPU Channel Interrupt Overflow Status Register for Engine 2 \(eTPU_GCR_CIOSR_ENG2\)](#). Interrupt Overflow status is not cleared automatically when Interrupt Status is cleared. The same mechanism and respective registers ([eTPU Channel Data Transfer Request Overflow Status Register for Engine 1 \(eTPU_GCR_CDTROSR_ENG1\)](#) and [eTPU Channel Data Transfer Request Overflow Status Register for Engine 2 \(eTPU_GCR_CDTROSR_ENG2\)](#)) are available for Data Transfer Requests.

If interrupt is set and cleared at the same time, set prevails and overflow is not altered (keeps the same state as it was before, asserted or not).

Global Exception has no overflow status.

37.15.2.3 Parameter Access

37.15.2.3.1 Parameter Access Widths

From the Host side the SDM address space is mapped in bytes, and each 32-bit parameter occupies 4 contiguous, aligned bytes. The Host can read/write the SDM by 8-, 16-, or 32-bit accesses in aligned addresses.

In 32-bit access, Host can access all 32 bits or only the lower 24 bits with an automatic sign extension (see [Parameter Sign Extension area](#)).

37.15.2.3.2 Parameter Addresses and Endianness

To access parameter number xxx, eTPU Microengine(s) would select address xxx. The Host would add $(xxx * 4)$ to the SDM base address to access the same parameter. For example, parameter 0x101 is seen by the Host in $(SDM\ base\ address + 0x404)$. An example of SDM memory map is shown in [Figure 37-298](#). The Host can access the SDM with a 32-bit-wide bus cycle to a four-byte aligned address, 16-bit-wide bus cycle to a two-byte aligned address, or 8-bit wide bus cycle to any byte address.

The address of the 24-bit parameters and the most significant byte depends on the endianness of the MCU. For more details, see the [Endianness](#).

37.15.2.3.3 Parameter concurrency

Host accesses to parameters may occur in parallel with eTPU Microengine accesses. Readings taken from a group of parameters while they are being simultaneously updated may lack coherency. eTPU provides mechanisms to ensure parameter coherency in accesses from both Host side and Microengine side, including the use of a coherent dual-parameter transfer mechanism, described in detail on [Parameter sharing and coherency](#).

37.15.2.3.4 Parameter Sign Extension area

The SDM address space to the Host is mirrored in a Parameter Sign Extension (PSE) area (see [Table 37-4](#)). Accesses from the Host to the PSE area differ from accesses to the standard SDM address space as follows:

- **Writes:** the most significant byte of the parameters is not written, and the SDM retains the old byte value, regardless of the Host access size.
- **Reads:** the most significant bit of the 24-bit parameter (that is, the MSB of the second most significant 32-bit parameter byte) is repeated in the 8 most significant bits of the read value on all 32-bit reads and most significant 16- and 8-bit reads.

The same parameters written in the standard SDM address space are read from the PSE area with the same offsets, and vice-versa. See [Table 37-429](#) for a reference of the address offsets in big and little endian machines.

This feature relieves the Host from extending the signal of 24-bit eTPU parameters before calculations, and from read-modify-write accesses to modify 24-bit parameters at the SDM.

37.15.2.4 SDM organization

The SDM internal partition for channel allocation is dynamic and programmed in the Channel Registers (see [eTPU Channel n Configuration Register for Engine 1 \(eTPU_CCCR_CnCR_ENG1\)](#) and [eTPU Channel n Configuration Register for Engine 2 \(eTPU_CCCR_CnCR_ENG2\)](#)).

The Host application is responsible for allocating a different parameter base address to each channel during the initial eTPU configuration, and to allocate enough parameters for the selected Function, with no unintentional overlapping between parameters of different functions.

Besides channel parameters, global areas may have to be allocated for parameters that are shared by more than one channel, in one or both Engines. Also, temporary parameter areas should be reserved to be used by the coherent parameter transfer mechanisms described in [Parameter sharing and coherency](#), if necessary. An SDM organization example is shown in the following figure.

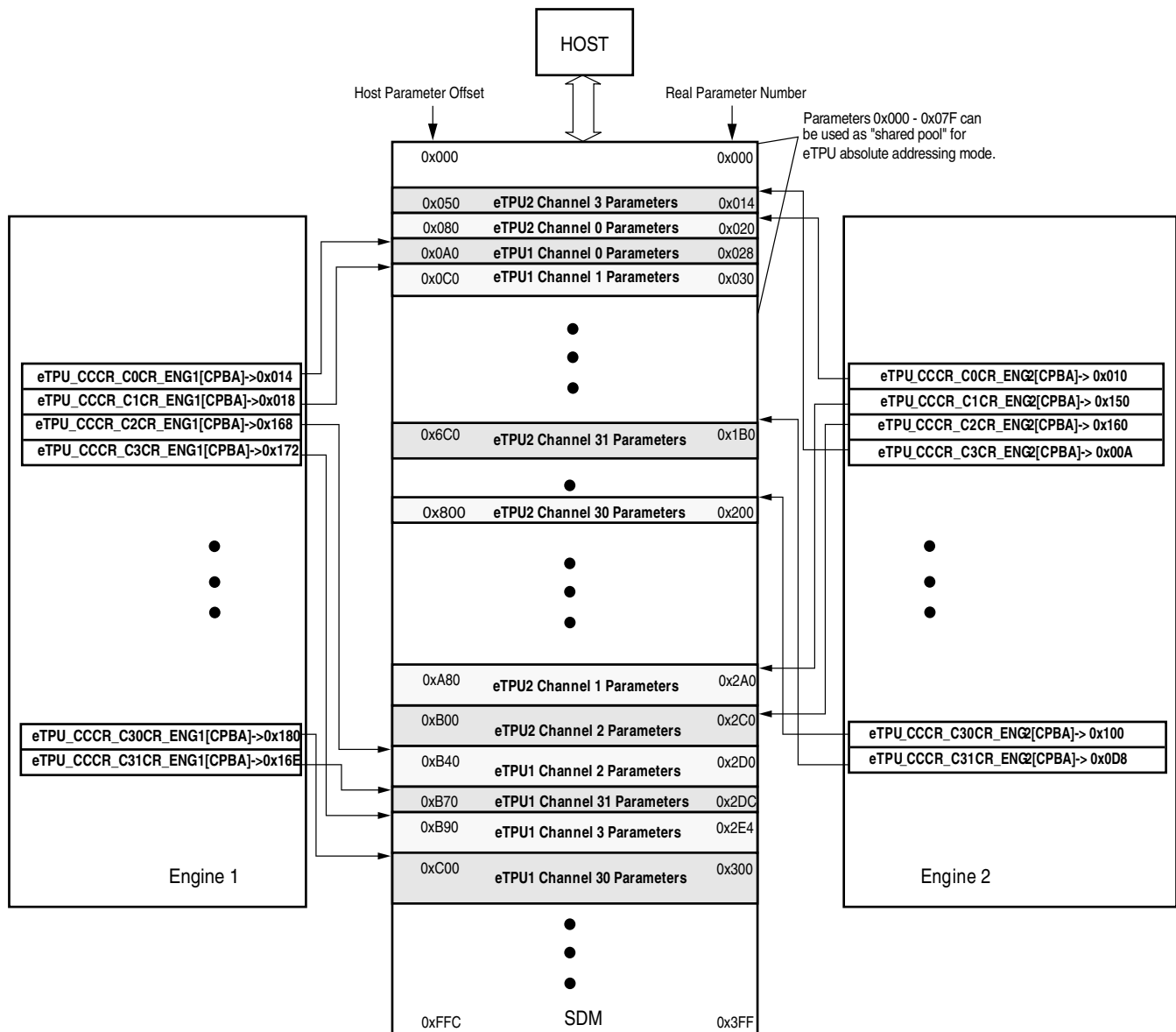


Figure 37-298. SDM Organization Example (4 KB)

Single-engine eTPU or dual eTPU system may require less parameters than the maximum number provided by the SDM. Since the SDM partition is fully dynamic, there is no limitation of fixed channel addresses, and the reduced array can be fully utilized.

37.15.2.5 Host Service Requests

Host CPU can request immediate service from a channel by writing a non-zero value to the Host Service Request register field - HSR (see [eTPU Channel n Host Service Request Register for Engine 1 \(eTPU_CCCR_CnHSRR_ENG1\)](#) and [eTPU Channel n Host Service Request Register for Engine 2 \(eTPU_CCCR_CnHSRR_ENG2\)](#)). There is one HSR field for each channel, so that writing to it generates a Service Request to the respective channel only. A zero value in HSR means no Host Service Request is pending for the channel.

HSR value turns to 000 automatically at the end of microengine service for that channel, but only if the thread started due to an HSR.

The meaning of a non-zero HSR value depends on the Function assigned for the channel. These bits are part of the conditions which select the Function entry point, and cannot be tested by microcode. For more details, refer to [Entry Points](#).

If Host writes HSR=000 when a thread for the same channel is already running, the thread runs until the end and is not aborted. If Host writes HSR>000 when an HSR thread for the same channel is already running, HSR value resets at the end of the thread, and no new HSR will be pending. If HSR is written before its value is resolved by the scheduler during TST, the entry point will obey the new HSR value, and if this new value is 000, no service thread is executed for the HSR.

The scheduling of HSRs is completely asynchronous with Host accesses, and there is no race-free manner to change an HSR value before service thread execution, so generally the safe way is: write HSR>0 only when HSR=0. Error recovery or emergency host procedures may require one to safely abort service and reset channel state when an HSR is already pending or executing. In these cases, the procedure below should be followed:

1. Disable the channel, writing CPR=00 in [eTPU Channel n Configuration Register for Engine 1 \(eTPU_CCCR_CnCR_ENG1\)](#) and [eTPU Channel n Configuration Register for Engine 2 \(eTPU_CCCR_CnCR_ENG2\)](#). That will prevent any pending HSR to be serviced.
2. Check if the channel is currently being serviced, reading its service status bit in [eTPU Channel Service Status Register for Engine 1 \(eTPU_GCR_CSSR_ENG1\)](#) and [eTPU Channel Service Status Register for Engine 2 \(eTPU_GCR_CSSR_ENG2\)](#). If

it is, wait for the time necessary to finish the service pending, or check again until $HSR == 0$, or channel service bit in [eTPU Channel Service Status Register for Engine 1 \(eTPU_GCR_CSSR_ENG1\)](#) and [eTPU Channel Service Status Register for Engine 2 \(eTPU_GCR_CSSR_ENG2\)](#) is cleared.

3. Write HSR with the error recover value. This value should, possibly combined with other host-defined flags in SDM or FM bits, initiate a channel reset or error recovery procedure.
4. Re-enable the channel, writing CPR value > 0 in [eTPU Channel n Configuration Register for Engine 1 \(eTPU_CCCR_CnCR_ENG1\)](#) and [eTPU Channel n Configuration Register for Engine 2 \(eTPU_CCCR_CnCR_ENG2\)](#).

37.15.2.6 SCM access

Only Host can access SCM as data. Depending on the specific device, SCM may be implemented as a RAM or ROM. This determines Host accesses to the SCM as shown below.

37.15.2.6.1 SCM RAM implementations

When SCM is implemented as RAM, the Host may read or write to SCM by setting [VIS](#) = 1. If $VIS=0$ and Host tries to access SCM space, a bus error is issued, writes are ineffective and read data is meaningless. Both Engines must be stopped or halted to set $VIS=1$.

Note

It is necessary to turn [VIS](#) bit on to set software breakpoints (see [Software breakpoints](#)).

37.15.2.6.2 SCM low power

SCM turns off its internal clocks when both Engines are stopped (STF field in [eTPU Engine 1 Configuration Register \(eTPU_SCR_ECR_ENG1\)](#) and [eTPU Engine 2 Configuration Register \(eTPU_SCR_ECR_ENG2\)](#) asserted), [VIS](#), and [MISC](#) is not enabled ($SCMMISEN=0$). The SCM clocks are automatically turned on if either one of the STF bits is negated or [VIS](#) turns to 1, or [SCMMISEN](#) turns to 1.

SCM clocks are not turned off if any of the Engines is not stopped, even if they are both halted.

The conditions for SCM Clocks and MISC activation are summarized in the following table.

Table 37-335. SCM clocks and MISC activation

STF in eTPU_SCR_ECR_ ENG1	STF in eTPU_SCR_ECR_ ENG2	VIS in eTPU_SCR_MCR	SCMMISEN in eTPU_SCR_MCR	SCM clocks	MISC
0	x	0 ¹	1	On	On
0	x	0 ¹	0	On	off
x	0	0 ¹	1	On	On
x	0	0 ¹	0	On	off
1	1	0	0	off	off
1	1	0	1	On	On
1 ²	1 ²	1	0	On	off
1 ²	1 ²	1	1	On	off ³
0	0	x	0	On	off
0	0	x	1	On	On

1. VIS cannot be written 1 if eTPU_SCR_ECR_ENG1[STF]=0 or eTPU_SCR_ECR_ENG2[STF]=0, and both HLTF bits are 0.
2. If VIS=1, neither MDIS can be written 0 nor the Engine leaves Stop Mode, regardless of device stop request.
3. MISC resets and stays so when VIS=1, restarting automatically when VIS goes 0 if SCMMISEN=1.

37.15.2.6.3 SCM Off-range data

When read accesses are made, either by the Host or by a microengine, to addresses above the limit corresponding to [SCMSIZE](#), the value read comes from [eTPU SCM Off-range Data Register \(eTPU_SCR_SCMOFFDATAR\)](#). The Host can program the register at initialization with an opcode value with operations that try to protect or recover the system from runaway code, for instance: terminate the thread, clear channel flags, disable match and transition service requests, issue an interrupt, jump to an error recovery procedure¹³. Writes to unimplemented addresses do not return error and can write on unspecified mirror addresses, so they should be avoided.

37.15.2.7 Bus error conditions

Follows a summary of the possible causes of bus errors on host accesses to the eTPU memory space:

- Read or write access to the SCM with [VIS](#) =0.

13. Only part of these suggested operations can be parallelized in a single instruction, see [Microinstruction formats](#).

- Read or write access to the channel registers with eTPU_SCR_ECR_ENG n bits MDIS=1, or bit STF=1 (see [eTPU Engine 1 Configuration Register \(eTPU_SCR_ECR_ENG1\)](#) and [eTPU Engine 2 Configuration Register \(eTPU_SCR_ECR_ENG2\)](#)).
- Write access to the Time Base Registers with eTPU_SCR_ECR_ENG n bits MDIS=1, or bit STF=1 (see [eTPU Engine 1 Configuration Register \(eTPU_SCR_ECR_ENG1\)](#) and [eTPU Engine 2 Configuration Register \(eTPU_SCR_ECR_ENG2\)](#)).
- Non-correctable errors in the SCM on reads of any size or writes less than 32-bits wide when CEDD =0.
- Non-correctable errors in the SDM on reads when DEDD =0.
- Accesses to the Shared Subsystem Area, when propagating a bus error issued by the attached subsystem.

37.15.3 Scheduler

Every Function is composed of one or more Threads. A Thread consists of a group of instructions that, once begins execution, cannot be interrupted by host or channel events. Each active channel intends to be serviced, being granted time for Thread execution. Since one microengine handles several channels operating concurrently, the Function threads must be executed serially.

The task of the Scheduler is to recognize and prioritize the channels needing service and to grant execution time to each channel. The time given to an individual Thread for execution or service is called a **Time Slot**. The duration of a time slot is determined by the number of instructions executed in the Thread plus SDM wait-states received, and varies in length.

At any time, an arbitrary number of channels can require service. To request service, channel logic, eTPU microcode or Host application notifies the Scheduler by issuing a Service Request.

37.15.3.1 Channel enabling and priority assignment

Every channel is assigned one of three priority levels - high, middle, or low - by the Host CPU, through the Channel Configuration Register field CPR (see [eTPU Channel \$n\$ Configuration Register for Engine 1 \(eTPU_CCCR_C \$n\$ CR_ENG1\)](#) and [eTPU Channel \$n\$](#)

[Configuration Register for Engine 2 \(eTPU_CCCR_CnCR_ENG2\)](#)). These registers are also used to disable the channel, which is equivalent to assigning it a "null" priority. In this case, the Scheduler does not grant any of its Service Requests.

It is possible to change the channel priority level or disable it dynamically. If the Host disables a channel when it is currently being serviced, channel service thread will complete. This means that it is possible for the output level of a channel signal to change, or a Host interrupt occur, even after its priority register was written to "null". For instance, if an output transition is scheduled, the transition will occur even after the channel is disabled.

Service requests previously pending or that occur while a channel is disabled remain asserted while the channel is disabled, and are serviced if the channel is enabled again, in due time determined by the priority scheme and concurrent requests from other channels. Channels are disabled after reset, and it is recommended to configure a Host Service Request for initialization of a channel before that channel is enabled to active priority (see [Initialization/application information](#)).

37.15.3.2 Channel priority schemes

The Scheduler holds a Service Grant register with one bit for each channel. Once the Scheduler grants a time slot to channel, the Service Grant bit for that channel is asserted in the Service Grant register. When the Service Grant bit of a channel is set, the channel may request new service but is not serviced again before its Service Grant bit is cleared.

When all channels in a same priority level are serviced, their Service Grant bits are cleared at the end of the thread, one eTPU clock before the next serviced channel is calculated, according to the scheme below¹⁴:

- Clear all grant bits of priority High if all channels of that priority that are requesting have their grant bits in 1.
- Clear all grant bits of priority Medium if all channels of that priority that are requesting have their grant bits in 1.
- Clear all grant bits of priority Low if all channels of that priority that are requesting have their grant bits in 1.
- Clear all grant bits of disabled channels.

14. Grant bits are also cleared in the next clock, when the service channel is chosen, or when the microengine is idle, using the same scheme.

This scheme assures that no channel is left with its grant bit forever asserted (preventing it from being serviced again), even if the channel priorities are reassigned during the execution.

Priority level is determined based on the maximum latency desired for each channel. A channel having a Function that requires the most frequent or more immediate service should be allocated a high priority level.

The eTPU employs a **primary** and a **secondary priority scheme**. These two schemes ensure frequent servicing of high-demand Functions and ensure a minimum time allocation to all channels requesting service, regardless of their priority level. The primary scheme prioritizes requesting channels that have different priority levels; the secondary scheme prioritizes requesting channels that have the same priority level.

Initially, a channel requests service and is granted a time slot by the Scheduler: Service Grant bit is asserted. If only high-level channels constantly receive service first because of their priority level, middle- and low-level channels would only be serviced by default, i.e., if no high-level channels request service. To ensure that each priority level receives an opportunity for servicing, every time slot has a fixed priority level that the Scheduler honors first. Divided into sets of seven, time slots are numbered from one to seven. The figure below illustrates the numbered time slots in sets of seven (fields A and B) and identifies their assigned default priority level. The high level has more time slots than the middle and low levels. Out of every seven time slots available, four are assigned to honor high-level channels first, two are assigned to honor middle-level channels first, and one is assigned to honor low-level channels first. Only one request (in each Engine) is serviced per time slot. When no channel requests service and the microengine is idle the priority scheme is initialized to time slot one, to prevent priority inversion on the next request¹⁵.

15. Priority inversion would occur in the following situation: no channel is requesting service, and the current time slot is primarily assigned to a low-priority channel. If the Scheduler was not reset to time slot one and two channels requested service at the same time, one with high priority and the other with low priority, the channel to be serviced would be the low-priority channel.

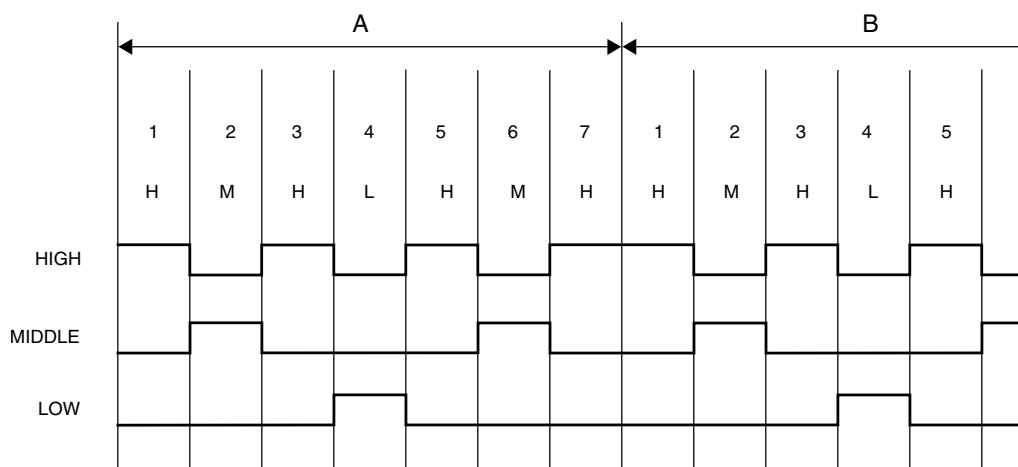


Figure 37-299. Time Slot Priority levels

37.15.3.2.1 Primary Scheme - Priority Among Channels on Different Levels

Although time slot priority assignment is fixed, the servicing priority is not. The primary scheme acknowledges the priority level assigned to a time slot, granting service first to a channel having the same priority. In [Channel priority schemes](#), time slot one has a high-level assignment; therefore, a high-level channel requesting service is recognized first. However, if no high-level channel requests service, the Scheduler recognizes a requesting middle-level channel. If this level has no request, the Scheduler continues to the low-level. If no requests occur, the Scheduler truncates the seven state cycle and starts a new cycle at time slot one, waiting for the first request. Granting service to a different-level channel is called priority passing. The order of passing always gives the highest priority to the assigned level, and the second priority to the higher of the remaining requesting priority levels as shown in the following table.

Table 37-336. Priority Passing

Assigned Priority Level	Next Priority Level	Next Priority Level
High	Middle	Low
Middle	High	Low
Low	High	Middle

When priority is passed to another level, that level is serviced and the fixed-priority-level sequence is resumed with the next time slot.

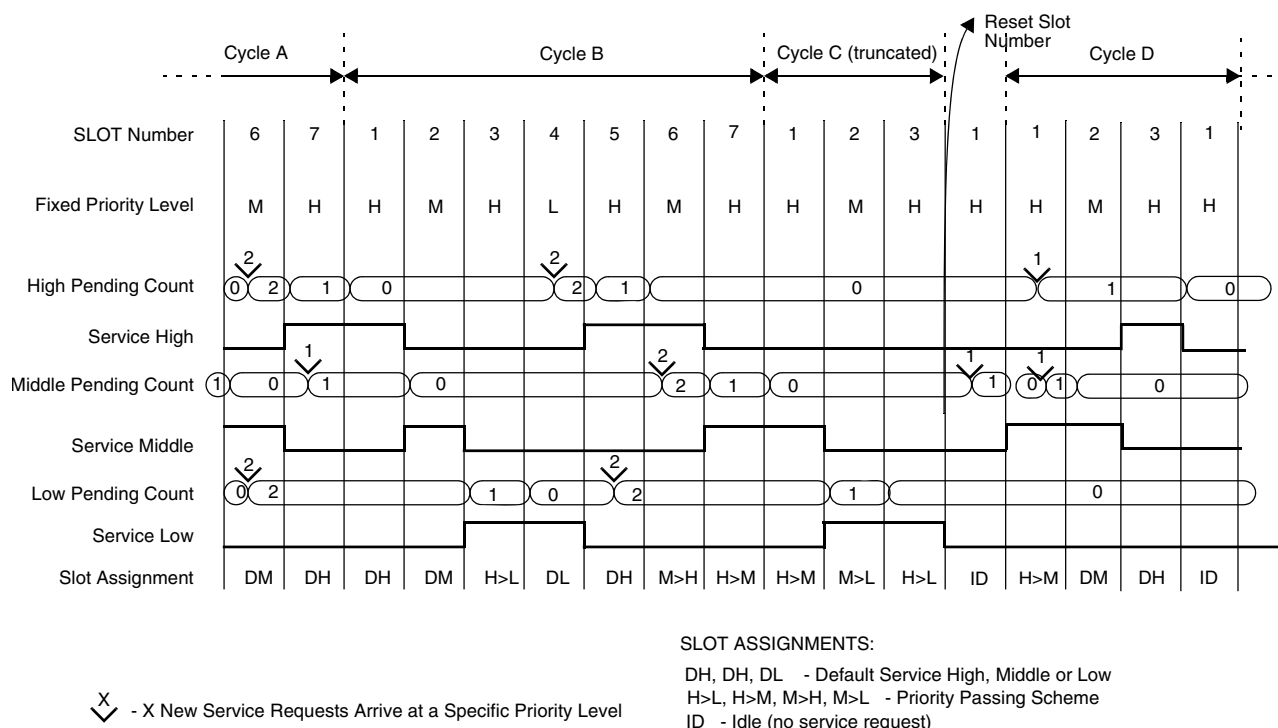


Figure 37-300. Priority Passing Example

Examples of priority passing are shown in the above figure. Each cycle contains seven time slots (or less if no service request exist). In cycle B, no high-level or middle-level service requests are present before time slot three which is assigned by default to high-level priority. Thus, time slot three is passed to the low level. In cycle B there are also no middle-level service requests before time slot six, so it passes the priority to a requesting high-level channel. During time slot six no more high level requests are left, but two new middle-level requests arrive, and there are also three low level pending service requests. Thus, time slot seven of cycle B and time slot one of cycle C are passed to the middle-level which is the next priority level after high. Time slots two and three of cycle C are passed to the low level which contains the three remaining channel service requests. At time slot three of cycle C the last low level request is serviced, and the Scheduler passes to idle state. At this point the cycle C is truncated and the Scheduler passes to time slot one of cycle D.

37.15.3.2.2 Priority passing disabling

The priority passing scheme allows a case where a high priority channel loses to a lower priority one right after another lower priority has been serviced, exemplified in the Cycle D on [Figure 37-300](#). A middle priority channel wins time slot 1 due to priority passing from high to middle. While it is being serviced, two new service requests arrive, one high and one middle priority. The high priority request loses to the middle one on next time slot 2 by default priority assignment.

This priority inversion can be avoided by setting the SPPDIS field in the eTPU_SCR_ECR_ENG n registers (see [eTPU Engine 1 Configuration Register \(eTPU_SCR_ECR_ENG1\)](#) and [eTPU Engine 2 Configuration Register \(eTPU_SCR_ECR_ENG2\)](#)), which disables the priority passing mechanism. When priority passing is disabled, at the end of the thread the slot number is incremented until a time slot that matches the priority of one of the requesting channel(s). The time slot advance takes no extra clocks. If no channel requests service, the time slot counter stays at time slot 1. The priority selection scheme with disabled priority passing is summarized in the following table.

Table 37-337. Priority Passing Disabling

At the end of time slot	servicing priority	if any request of priority	service it on time slot	else if any request of priority	service it on time slot	else if any request of priority	service it on time slot
1	High	Medium	2	High	3	Low	4
2	Medium	High	3	Low	4	Medium	6
3	High	Low	4	High	5	Medium	6
4	Low	High	5	Medium	6	Low	4
5	High	Medium	6	High	7	Low	4
6	Medium	High	7	Medium	2	Low	4
7	High	High	1	Medium	2	Low	4

An example of the priority passing disabling scheme is illustrated in the figure below. The sequence of service requests is the same as in the example of [Figure 37-300](#), and although the time slot incrementing differs, the priorities granted are the same for cycle B. Cycle C has one of the low priority channels serviced before the second middle one. Cycle D, however, no longer has the priority inversion.

In cycle B, after the time slot 2 only a low priority request remains, so the time slot count advances directly to 4, which has a low priority assigned. Time slot keeps on 4 for the next service, as only a low priority request remains also, and only time slot 4 is assigned to low. Two high priority services contend for the next time slot 5 (assigned to High). The second high priority channel is serviced on the next time slot, jumped to 7 because there is no middle request, ending cycle B. Cycle C starts with time slot 2, as there are no high priority requests and two middle and two low ones. After the first middle service, time slot count skips 3 assigned to high (no high requests), and services a low priority channel on time slot 4. It follows the same scheme until there are no other requests and cycle C is truncated, resetting the time slot counter to 1.

Cycle D begins with a middle request, jumping to time slot 2. During this service two requests arrive, one high and one middle. Unlike what happened with priority passing, the next serviced is the high priority channel, as the time slot increments to 3. The second middle priority channel request in cycle D is finally serviced next, on time slot 6.

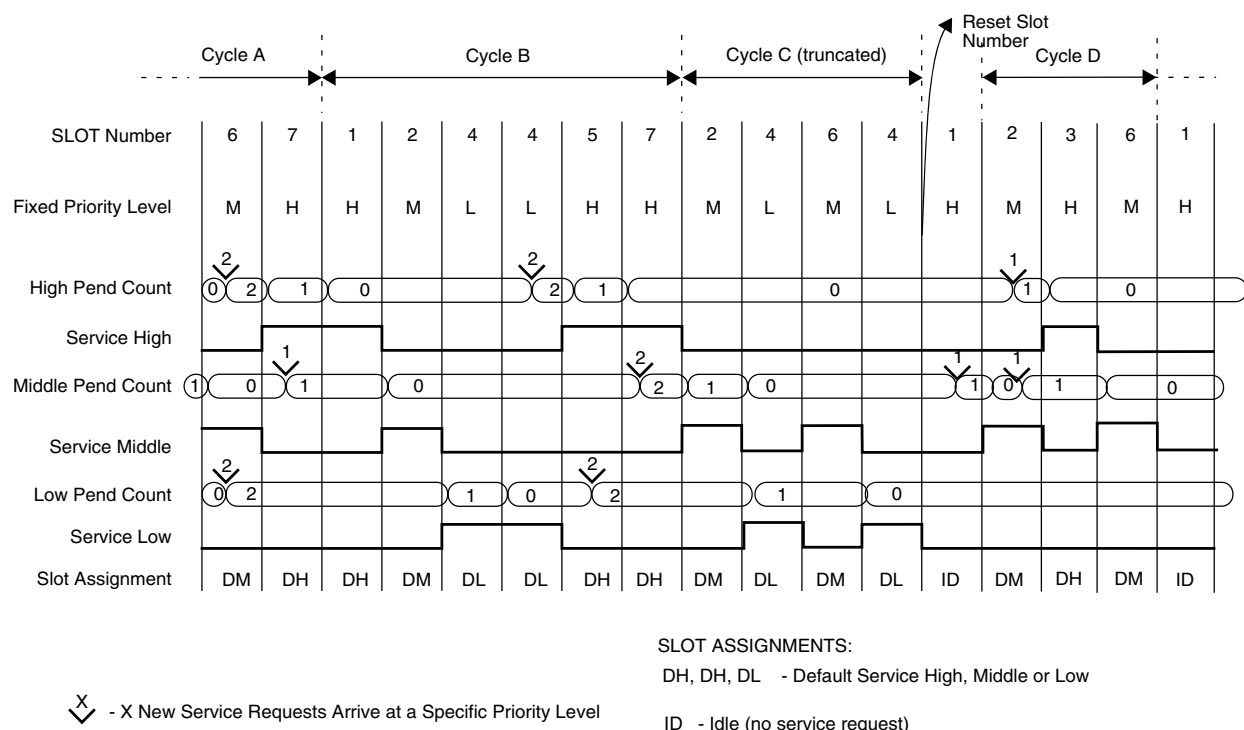


Figure 37-301. Priority Passing Disabling Example

37.15.3.2.3 Secondary Scheme - Priority Among Channels on the Same Level

Because channels can randomly request service, channels having the same priority level will inevitably request service simultaneously. A secondary scheme prioritizes these requests. The Scheduler services channels on each of the three priority levels, beginning with the lowest numbered channel on that level.

37.15.3.2.4 Priority Scheme Example

The overall priority scheme simultaneously incorporates both primary and secondary schemes. Combining both schemes in the following example conveys their correlation.

1. One high-priority and one low priority channels request service, while the Scheduler is in time slot one. Having its service request bit asserted, a single high-level channel is granted the time slot, which has high-level priority (primary scheme) and its service grant bit is asserted. At the end of the thread, the service grant bit is negated (no more requests of high priority level channels).

2. The Scheduler proceeds to time slot two, which has middle-level priority; however, no middle-level channel is requesting service. Priority is passed to the high level, but no high-level channel is requesting service; therefore, priority is passed again, and service is granted to the single requesting low-level channel. Once serviced, this channel's grant bit is negated (no more low-level requests).
3. The Scheduler resumes with the fixed-priority sequence on time slot three; however, no channels are requesting service. The Scheduler returns to time slot one, waiting for requests.
4. Two high-level and two middle-level channels simultaneously request service. Being in time slot one which is assigned high priority, the Scheduler finds the lowest numbered high-level channel (secondary scheme) and selects it for service. This channel's service grant bit is asserted.
5. The Scheduler continues to time slot two, which has middle priority (primary scheme), and allocates the slot to the lowest numbered middle-level channel requesting service (secondary scheme). The Scheduler notes the still unserviced middle-level channel and proceeds to time slot three.
6. Time slot three is allocated for high priority. The slot is allocated to the remaining unserviced high-priority channel, and the channel's service grant bit is asserted. The Scheduler checks again at the end of the thread. All service grant bits of high-level requested channels are asserted; therefore, all high-priority channels that requested have been allocated execution time. Under this condition, all service grant bits of the high-level serviced channels are negated. The Scheduler proceeds to time slot four.
7. Time slot four is allocated for low-priority channel; however, no low-level channel is requesting service. Priority is passed to the high level, but no high-level channel is requesting service; therefore, priority is passed again, and service is granted to the remaining middle-level channel which requests service. This channel's service grant bit is asserted. The Scheduler checks again at the end of the thread. All grant bits of middle-level requested channels are asserted; therefore, all middle-priority channels have been allocated execution time. Under this condition, all service grant bits of the middle-level serviced channels are negated. The Scheduler proceeds to time slot five. Meanwhile a low priority channel requests service.
8. Time slot five is allocated for high-priority channels, but there are no more requests from high-priority or middle priority channels. The single low-level channel which required service is granted time slot five. Once serviced, the channel's service grant bit is asserted. Next, the service grant bit is negated (no more requests of low priority level channels).

9. The Scheduler resumes with the fixed-priority sequence on time slot six; however, no channels are requesting service. The Scheduler returns to time slot one and waits for requests.

37.15.3.3 Time slot latency

Latency is the amount of time between a service request and the beginning of service on that channel. The following factors affect latency:

- Number of active channels
- Number of channels on a priority level
- Number of available time slots on a priority level
- Number of microcycles required to execute a thread of a Function
- Number of SDM accesses during execution of a Function thread
- eTPU clock frequency

Each time slot may require a different number of microcycles, depending on the thread of a Function to be executed. This variation is shown in the figure below.

For more details on latency evaluation, see [Estimating Worst Case Latency](#).

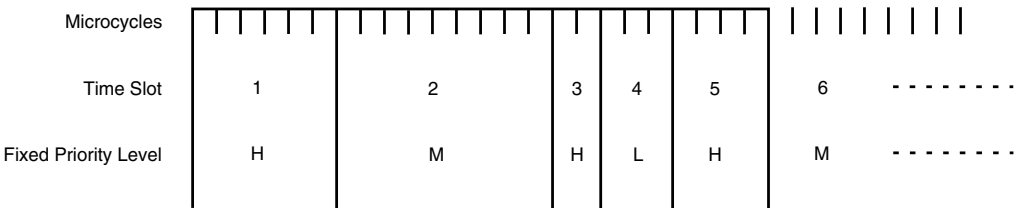


Figure 37-302. Time-slot variation

37.15.4 Parameter sharing and coherency

SDM can be concurrently accessed by Host and Microengines (two in a Dual eTPU Engine system). In general, there is no guaranteed order by which a group of parameters is accessed, which may lead to a lack of internal consistency if two or more related parameters are read when only part of them is updated.

eTPU provides mechanisms to guarantee parameter coherency. The most generic mechanisms for Host-eTPU coherency, suitable for any number of parameters, are:

- the use of Transfer Service Thread mechanism.
- the mailbox (or "software semaphore") mechanism.

These mechanisms, described in [Multiple parameter coherency methods](#), use microcode to transfer parameters from temporary buffers in SDM to their definitive locations (or vice-versa). These methods have the disadvantage of wasting processing and code memory resources.

eTPU also provides a **Coherent Dual-parameter Controller - CDC** - mechanism. It is used by Host to coherently transfer pairs of parameters from/to a parameter buffer located on SDM to/from the locations on SDM where parameters are accessed directly by the channels. Coherency is guaranteed by SDM access arbitration. Although limited to two parameters only, it has lower latency and wastes no microengine resources¹⁶. CDC usage is described in [Coherent Dual-parameter Controller \(CDC\)](#).

For parameters shared by both Engines, eTPU provides **Hardware Semaphores**. Coherency is assured given the semaphores are used to prevent concurrent access to the changing parameters. Microengine can request semaphores using specific microinstructions (see [Semaphore Operations](#)). Hardware Semaphores are described in detail in [Hardware Semaphores](#).

Neither Host nor CDC have access to the hardware semaphores, but they can be combined with microcode transfer mechanisms if Host must coherently access parameters which are also shared by both Engines.

In order to ensure coherent access to a group of parameters by two or more contenders, **each contender must have atomic access** to the shared parameters. Atomicity conditions are discussed in [Host side atomic access](#) and [Microengine Side Atomic Accesses](#).

37.15.4.1 Host side atomic access

Host side atomic accesses can be achieved by either of following ways:

- For one parameter, the SDM should be accessed by 32-bit-wide data transfers to ensure coherency

16. A microengine access to the SDM in the moment CDC is performing the transfer may suffer a maximum of two wait-states.

- For two parameters only, using the Coherent Dual Parameter Controller.
- Indirectly, for any number of parameters, by requesting microcode to coherently access SDM in its behalf. The host side atomicity problem becomes, then, a microengine side atomicity problem. Some methods that use this approach to achieve coherency are described in [Multiple parameter coherency methods](#).

37.15.4.2 Microengine Side Atomic Accesses

37.15.4.2.1 Microengine Single Parameter Atomicity

SDM should be accessed by 32-bit-wide data transfers to ensure atomicity for 32-bit parameters. This applies either to Host-Microengine coherency or Microengine-Microengine coherency in a dual eTPU Engine system.

37.15.4.2.2 Microengine Dual Parameter Atomicity

Microengine has the ability to access two parameters coherently in back-to-back accesses, at random addresses: once it accesses SDM, it has priority over Host for another access in the next microcycle (see [SDM arbitration](#)). Note that it applies **only to Microengine-Host** coherency. For Microengine-Microengine coherency in a dual eTPU Engine system, one must use Hardware Semaphores (see [Hardware Semaphores](#)).

Microengine dual back-to-back accesses are guaranteed to be atomic in relation to Host slave accesses or Coherent Dual-parameter Controller, regardless of semaphore usage: Host or CDC accesses cannot break-up a back-to-back Microengine access, neither Microengine can break a CDC transfer, due to the SDM arbitration mechanism described in [SDM arbitration](#).

Atomicity is not guaranteed if microengine enters halt state in the middle of a back-to-back access (see [Microengine Halt State](#)): Host can access SDM while microengine is halted in the middle of a back-to-back access.

37.15.4.2.3 Microengine Side Multiple Atomicity

Hardware Semaphores must be used for Microengine-Microengine coherency (more than 1 parameter) since two or more accesses from one Microengine are not atomic with respect to the other.

For multiple Microengine-Host coherency, the software methods described in [Multiple parameter coherency methods](#), or similar ones, must be used. Some of these methods rely on the fact that parameter access of a thread is atomic in relation to another thread in the same Engine, since a thread cannot be suspended (preempted).

For 1 parameter coherent access, or dual parameter coherency between only one Microengine and Host, the alternatives shown in previous sections apply.

37.15.4.3 Coherent Dual-parameter Controller (CDC)

Dual-parameter coherency is supported by a Coherent Dual-parameter Controller hardware (CDC), which contends with microengine for SDM access. CDC atomically transfers, upon Host's command, two parameters from one area of the SDM to another. One area is a temporary (buffer) area, where the two parameters are directly read or written by the Host. This temporary area has to begin in an SDM address multiple of 2 words, and the two parameters must be sequential. The other area is the channel parameter area where the microcode normally accesses the parameters, usually with the channel relative address mode (see [SDM Addressing Modes](#)). In this area, the parameters transferred by CDC don't have to be sequential. A transfer from the temporary area to the channel area, when the Host sends data to the channel, is called a **write transfer**. Inversely, in a **read transfer** the parameters are copied from the channel area to the temporary area (channel to Host).

Coherency is guaranteed by the SDM access contention rules implemented in the SDM arbiter (see [SDM arbitration](#)). CDC transfers are coherent in respect to the two Engines, so the target parameters in the channel area may be shared by channels on them both. During CDC operation, the Host may suffer from 3 up to 11 eTPU clocks wait states¹⁷, and the Microengine(s) may suffer up to 2 microcycle wait-states¹⁸. CDC accesses are atomic with respect to Microengine(s) accesses to the SDM. Even when neither engine is in TST, CDC may suffer up to 4 eTPU clock internal wait-states from SDM arbiter, meaning 9 slave wait-states to Host, so that it does not break atomic back-to-back accesses from microengine(s). CDC also cannot break TST preload accesses. Host can initiate CDC back-to-back transfers: there is no need of idle slave cycles between two transfers.

-
- 17. The maximum number of Host wait states on CDC occurs when both microengines overlap their TSTs, delayed 3 eTPU clocks from each other.
 - 18. One microcycle takes two eTPU clocks. Microengines get wait-states in multiples of microcycles, while Host and CDC wait-states are multiples of eTPU clocks.

Note

SDM memory error detection and correction does not affect the number of wait-states to host due to CDC (see [Memory error support](#)).

CDC can be used to atomically fix SDM soft errors. For more details, see [Error fixing](#).

37.15.4.3.1 CDC programming

The Coherent Dual-parameter Controller Register (see [eTPU Coherent Dual-Parameter Controller Register \(eTPU_SCR_CDCR\)](#)) is used to configure and initiate CDC transfers between the temporary area and channel parameter area. Host asserts STS bit in order to start the data transfer. CDC then contends for the SDM and starts the transfer. When the data transfer is complete, STS returns to 0. Host receives wait-states for writing STS=1 while CDC contends for SDM and during the transfer. The write access ends when CDC finishes the transfer. Host receives wait-states during the CDC transfer. If Host writes [eTPU Coherent Dual-Parameter Controller Register \(eTPU_SCR_CDCR\)](#) with STS=0 or does not write the STS byte, the CDC transfer does not occur. CDC programming can be summarized as follows:

1. If it is a write transfer, i.e., from Host to channel, write the two parameters into temporary area.
2. Write **STS** =1 and the remaining CDC programming parameters: parameter width (32 or 24 bits, field PWIDTH), transfer direction (read or write, field WR), temporary parameter area base address (field PBBASE), and the absolute addresses of the parameters to be transferred (concatenation of the fields CTBASE and PARM0/1).
3. If it is a read transfer -- that is, from channel to host -- read the two parameters from the temporary area into Host memory/registers.

37.15.4.4 Hardware Semaphores

eTPU provides Hardware Semaphores accessible by the Microengine only. It is the responsibility of the application to ensure proper use of the semaphores (i.e., agree upon a specific semaphore and use it properly, to ensure coherency).

The eTPU microinstruction set has support for locking and freeing the semaphores, described in [Semaphore Operations](#), and this is the only way to access them.

There are four semaphores available, which reduces the amount of collisions by assigning unrelated data transfers to different semaphores. Semaphores are used for parameters which can be shared by channels in different Engines, and for Engine-to-Engine synchronization. Semaphores are also the only way to ensure coherent access to parameters shared between the two Microengines.

Attempting to lock one semaphore (even not successfully) frees the other locked by the same Engine, ensuring one can lock just one semaphore at a time. That prevents deadlock conditions between the two Engines.

Microcode END command or Engine being in idle state (no thread executing) automatically releases all semaphores from one Engine side, even if a semaphore lock is done in parallel. However, it is recommended to write the microcode in a way which locks semaphores for the shortest required period, and frees them without waiting for the END command, to improve the performance of the other microengine. Semaphores are free after reset. An Engine can only free a semaphore locked by itself.

Semaphore lock requests are always **non-blocking**, in the sense that they do not suspend the requester in case the semaphore is already locked. The semaphore status after the lock request - indicating if it was successfully locked or not - must be tested through the SMLCK microengine branch condition (see [Branch conditions](#)).

37.15.4.5 SDM arbitration

Up to four entities can access SDM:

- Two Microengines (in a dual eTPU Engine system)
- The Coherent Dual-parameter Controller (CDC)
- The Host CPU (direct memory-mapped access)

The following rules specify the access priorities for contended access. They keep compatibility with the TPU3 dual parameter access atomicity, but only between the microengine and CDC (not Host accesses through slave bus).

1. Microengine accesses from the two eTPU Engines are interleaved between each other, but not with Host or CDC accesses;
2. The eTPU microengine(s) gives priority for SDM accesses to either the Host CPU or the CDC under any of the following conditions:
 - a. The microengine has completed accessing the second parameter in a back-to-back SDM access¹⁹.

- b. The SDM was not accessed during the last arbitration slot for the microengine and the host does not lose the access to the other engine in the current arbitration slot²⁰
 - c. CDC is transferring data, after its first (read) access. Note that the CDC can be in middle of a data transfer of another pair of parameters, unrelated to the ones that microengine tries to access.
3. The eTPU microengine takes priority for SDM accesses under either of the following conditions:
- a. the Host CPU or CDC has done a data transfer during the last access arbitration slot for the engine²⁰. Also, the Host CPU does not hold a pending access against the other eTPU microengine.
 - b. the microengine is arbitrating for the access of its second parameter in a back-to-back access¹⁹. All pairs of back-to-back parameter accesses are coherent with respect to Host and CDC (not to the other microengine).

The direction (read or write) of any individual access by Host or microengine is irrelevant to the arbitration. The use of Normal or PSE SDM area by the Host is also irrelevant to the arbitration.

The first parameter preloading in a TST is considered first access by the arbiter, regardless of any access made at the END microinstruction of the previous thread, that is, the last access of a thread and the first preload are never considered a back-to-back access. On the other hand, the TST preload accesses are considered back-to-back and are, therefore, atomic with respect to Host or CDC.

Note

The Zero SDM operation (see [Zero SDM Operation](#)) is considered an SDM access for arbitration purposes both on writes and reads; the fact that read SDM data is discarded is irrelevant for arbitration.

19. If microengine tries to access the SDM in the following microcycles, the third and fourth consecutive accesses are considered the first and second of a new back-to-back dual access.

20. The microengine access slot is between its own T4 and T2 edges (see [Microcycle and I/O timing](#)).

37.15.5 Enhanced Channels

Enhanced Channels comprise hardware support for input digital signal processing and output signal generation. Each Channel is associated with one input and one output signal. Enhanced Channel logic is combined with Function microcode (and optionally Angle Mode logic) to implement Channel I/O functionality.

eTPU's Enhanced Channels are capable of **dual action**, meaning that each channel logic can handle two events at different times and/or cause two separated actions - these actions and events can be mutually dependent (with the first either blocking or enabling the other), or both independent, depending on the programmed **Channel Mode**.

Each Enhanced Channel contains event logic containing two **Event Register sets**, each set supporting one input and/or output action, the pair implementing dual action support. Each Event Register set contains two 24 bit registers: Match and Capture. The **Match register** holds the pending match value which is compared against one of the two time bases by an equal-only/greater-equal comparator. The **Capture register** captures one of the two time bases as a result of a Match or Transition detection. Service Requests are issued on particular combination of match and capture events, defined by the selected Channel Mode.

In the context of the eTPU channels, a **Match** is a comparison between a time base value and a channel Match register. If those two values are coincident, or the time base value is greater than the value of the Match register, a **Match Event occurs**. Depending on the channel mode of operation and current state of the channel logic, the match event may be recognized, i.e., change the state of the channel, or be ignored. A match event recognized by the channel logic is called a **Match Recognition**. Match Recognitions can cause, also depending on Channel Mode and current state, the channel to request service, configuring a **Match Service Request**.

eTPU uses two kinds of comparator to assert a Match Event: an **Equal comparator**, in which both the Match Register and the value of the selected time base must match exactly, and a **Greater-Equal comparator**. The Greater-Equal comparator considers any time base value between the range $[N : N+0x800000-1]$ as a valid match against the value of N in the Match Register, even when the value $N+0x800000-1$ wraps around the point of origin (0x0). See the following figure for an illustration of the matching values on a Greater-Equal comparator.

The second source of events for the eTPU channel is a **Transition** detected at the corresponding channel's input signal. Two distinct Transition detections can be programmed individually for each channel, allowing recognition of all possible combinations of edge detection. It is also possible to check the sampled state of an input

signal upon the occurrence of a Match: the sampling of the expected value is treated as a Transition, even if the input signal did not necessarily toggled at the time of the Match, or at any time at all.

Like Match Events, **Transitions Events** may or not be recognized by the channel logic. When they are, a **Transition Detection** occurs. As well as Match Recognitions, Transition Detections can issue a Channel Service Request, depending on Channel Mode and current state.

Transition Detections and Match Recognitions are sometimes simply called Transitions and Matches throughout this document, for short.

Input and output signals can be processed separately by the channel logic and microcode, and can also be combined such that Matches and Transitions are used to cause **output signal actions**. The output signals can also be directly controlled by microcode. Many event combinations are allowed for a channel, given the possibility of configuring pairs of matches and transitions for the dual action logic, where each event is able to block or enable the next one. There is a full set of Channel Modes described in [Channel Modes](#), exploring all the capabilities mentioned here.

Each channel has its own set of registers and flags. They are selected, and made accessible to the Microengine, according to the value written into the microengine **CHAN Register** that points to the desired channel . Every time the CHAN register is written, even if with the same previous value, a channel is selected and its flags and registers are updated. For further detail, see the Channel Selection Register-CHAN topic in the General Channel Registers section .

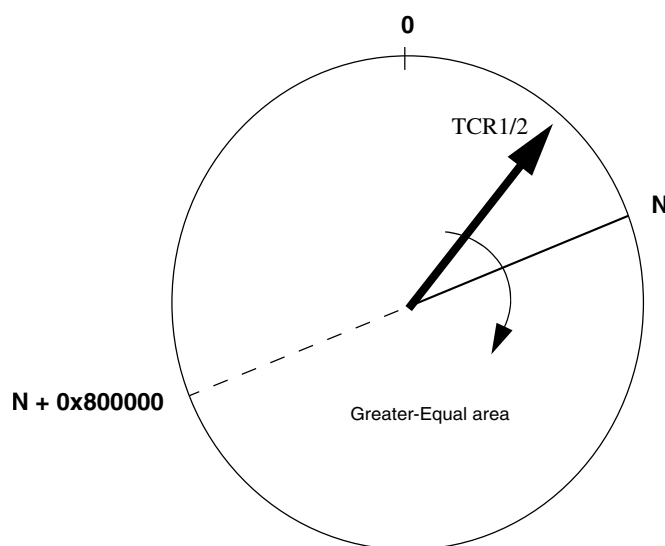


Figure 37-303. Greater-Equal Comparator

Beyond the request of services due to the signal and timing internal to each channel, one eTPU channel microcode can explicitly request service from another channel through the microengine **LINK Register**. A microcode write to the LINK Register asserts a service request to the channel whose number matches the contents of LINK. See [Channel Link](#) for a complete description of this mechanism.

These service requests originated in the eTPU Enhanced Channels (either time base match, input signal transition, or link service request) result in a call to the corresponding channel service routine, which is the sequence of microinstructions that is called a **Thread**. For further detail, see [Functions and threads](#).

In addition to Event Logic, each Channel has an Output Buffer Enable signal, controlled by microcode, and an Enhanced Digital Filter, which eliminates spurious glitches on input pin signal. See the chip-specific eTPU information for how the Output Buffer Enable signal is controlled.

A high level diagram of Channel logic and registers is shown in the following figure.

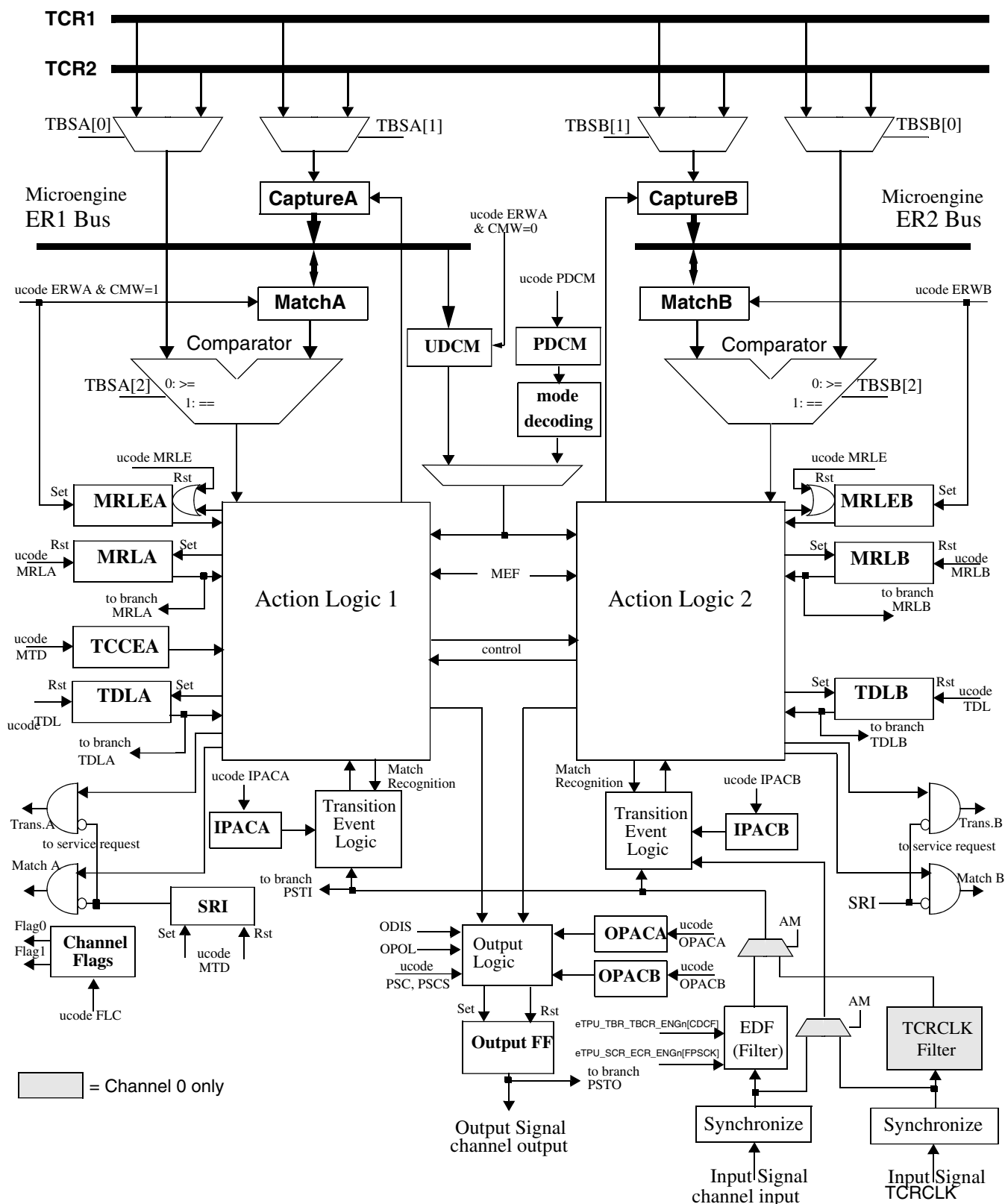


Figure 37-304. Channel logic block diagram

37.15.5.1 Channel registers and flags

Channel configuration and control registers can be divided in the following groups:

- **Host Configuration and Control registers**, which define channel Function and parameter allocation in SDM, input signal filtering, manage Host interrupts, and are used for Host Service Requests; they can only be accessed by Host, except for the Function Mode bits which can be also tested by microcode.
- **Event Registers**, which can only be accessed by eTPU Microengine, through dedicated Channel Control microinstruction operations (see [Channel Control and Configuration Microoperations](#)); these registers are directly used to implement channel functionality, and include channel event status latches which can be directly tested by Microengine branch instructions.
- **Pin Control registers**, which basically define pin state and transition polarity (but not input signal filtering); they are accessible only by dedicated Channel Control microinstruction operations.
- **Link registers**, which implement the channel link mechanism that allows one channel to request service to another one; they are accessible only by microinstruction operations.
- **General Channel registers**: CHAN, SRI, Flag0/1, PDCM, UDCM.

Most of those registers are channel exclusive, i.e., there is one copy of them for each channel. Microcode can access registers from only one channel at a time. The Channel Selection (CHAN) register (see [Channel Selection Register - CHAN](#)), accessible only by microcode, defines the channel whose registers are being accessed, with exception of link register and function mode. CHAN register assumes the value of the channel to be serviced at the beginning of TST.

The Service Request Inhibit - SRI - register controls the generation of Service Requests on matches and transitions, also affecting channel logic behavior. For a full description see [SRI - Match/Transition Service Request Inhibit Latch](#). Flag0/1 are used to select channel service threads based on channel software state. See [Flag1,Flag0 - Channel "state resolution" flags](#) for more details.

Host Configuration and Control registers are described in [Channel Configuration Control memory map and registers](#).

Time Base configuration is common to all channels, and described in [Time bases](#). Time Base selection for matches and captures, however, is individual to each channel, and is part of the Event Registers.

Link registers are described in [Channel Link](#).

The following sections describe the Event Registers and Pin Control registers.

37.15.5.1.1 ER - Event Registers

Each channel contains two identical Event Register sets, named ERA and ERB, corresponding to the two actions supported. Each Event Register set contains:

- a 24-bit Match register (Match A or Match B), which holds a match value. This value is compared against the selected match time base (TCR1 or TCR2).
- a 24-bit Capture register (CaptureA or CaptureB), which samples the selected capture time base (TCR1 or TCR2)
- a Time Base Selection register (TBSA or TBSB)
- a Match Recognition status flag (or latch) (MRLA or MRLB)
- a Match Recognition Enable latch (MRLEA or MRLEB)
- a Transition Detection flag (or latch) (TDLA or TDLB)
- a Transition Continuous Capture enable (TCCEA only)

ERA and ERB are associated with the first and second events in double action modes, not necessarily in that order. The order of Match events associated with ERA and ERB depends on the programmed channel mode, the MatchA and MatchB values, and the timebases selected by TBSA and TBSB. Similarly, the order of Transition events associated with ERA and ERB depends on the programmed channel mode, and the transition detection selected by IPACA and IPACB.

These registers are directly or indirectly accessed by the microcode. TBSA and TBSB registers are defined in [TBSA and TBSB - Time Base Selection Registers](#). The other registers are explained in [Match Recognition](#) and [Transition detection and Time Base Capture](#).

Access to the Event Registers is qualified by the channel currently selected by the microengine (i.e., the channel value currently in the CHAN register). During the channel transition period (automatic CHAN assignment), or whenever CHAN is written by microcode, Capture values of the new selected channel are sampled into Microengine registers ERTA and ERTB, therefore becoming visible to the microcode. At the same time, updated values of MRLA, MRLB, TDLA and TDLB are sampled into the branch logic, making the register values and the flags coherent with respect to each other and with the thread selected by the Scheduler. ²¹

21. The thread selected is determined by the Entry Point which, in turn, is determined partially by the channel latches. See [Entry Point address generation](#).

Note

The Function Mode bits are also sampled from the Host interface on Time Slot Transition, so that they remain constant to microengine even when Host changes them.

During service, the microcode can access updated values of the Event Registers of any channel by writing the channel number to CHAN. Writing CHAN with the same value (CHAN := CHAN) updates ERTA and ERTB with the new captured values, the branch logic with updated MRLA/B and TDLA/B flags. Writing CHAN with a different value does the same with the values from the newly selected channel.

Match values are also accessed through ERTA and ERTB Microengine registers, which are copied to/from the channel MatchA and MatchB registers by specific microinstruction operations.

Microcode writes to the flags and selections (MRLA/B, TDLA/B and TBSA/B) are immediately effective to the channel. The MRLA/B and TDLA/B branch conditions are also immediately reset when their correspondent flags are reset by microcode. Match registers are indirectly written by microcode through ERTA/B. MRLEA/B is unconditionally asserted when respective Match register is updated from ERTA/B, and its negation is immediate.

The following table summarizes Event Registers accesses.

Table 37-338. Event Registers microcode accesses

Register	Access type	Sampled from channel	Update to channel	Microcode fields ¹	Reset value ²
CaptureA, CaptureB	read through ERTA/B	to ERTA/B on CHAN assignment	no	T2ABD	n.a.
MatchA, MatchB	read and write through ERTA/B	to ERTA/B by microcode	from ERTA/B by microcode	ERWA, CMW, ERWB, T4ABS	n.a.
MRLEA, MRLEB	write to 0 (negate) directly; write to 1 (assert) upon MatchA/B update from ERTA/B	no	immediate	MRLE, ERWA, ERWB	0, 0
TBSA, TBSB	write only	no	immediate	TBSA, TBSB	000, 000
MRLA, MRLB	flag test on branch, write to 0 (negate) only	on CHAN assignment	immediate	BCC (test) MRLA, MRLB (reset)	0, 0
TDLA, TDLB	flag test on branch, write to 0 (negate) only	on CHAN assignment	immediate	BCC (test) TDL (reset)	0, 0
TCCEA	write only	no	immediate	MTD	0

1. See [Microinstruction Set](#).
2. n.a. means that value of the register is undetermined after reset.

37.15.5.1.1.1 MatchA and MatchB Registers

MatchA and MatchB registers hold a match value to be compared against the selected channel time base. A match value can only be written into the Match register by microcode, through ERTA/B microengine registers (see [Write Channel Match and UDCM Registers](#)). Microcode can also read the Match register as a special T4ABS source operation, when T4ABS=0101, and the source for T4ABS is selected from the second register set. In this operation, MatchA/B registers are copied into ERTA/B registers (see [Selecting sources and destination](#)). For more information on time base matches, see [Match Recognition](#).

37.15.5.1.1.2 CaptureA and CaptureB Registers

CaptureA and CaptureB registers capture the selected channel time base. CaptureA/B registers cannot be directly written or read by microcode. During the Time Slot Transition (TST) or during CHAN assignment, CaptureA/B registers are copied into ERTA/B microengine registers. For more information, see [Transition detection and Time Base Capture](#).

37.15.5.1.1.3 TBSA and TBSB - Time Base Selection Registers

TBSA/B are 3-bit registers which have the following effect on channel configuration:

- Selection of the timebase (TCR1 or TCR2) to be compared against the match values in MatchA and/or MatchB registers.
- Selection of the timebase (TCR1 or TCR2) to be captured in the CaptureA and/or CaptureB registers by a match or transition detection event.
- Selection of comparator mode to be used with MatchA and MatchB registers: equal-only or greater-equal.

After reset TBSA/B are 000. The following table shows values of TBSA and TBSB for configuration selection. Note that the time base selection for capture is independent of the time base selected for matches.

Table 37-339. TBSA/B Programming States

TBS bit	2	1	0
bit value	Comparator selection	Capture time base selection	Match time base selection
0	greater or equal	TCR1	TCR1
1	equal-only	TCR2	TCR2

TBSA/B are written through the microcode fields TBSA/B (see [Comparator and Time Base Selection](#)).

37.15.5.1.1.4 MRLA/B - Match Recognition Latches

See [MRLA/B - Match Recognition Latches](#).

37.15.5.1.1.5 MRLEA/B - Match Recognition Latch Enable

See [MRLEA/B - Match Recognition Latch Enable](#).

37.15.5.1.1.6 TDLA/B - Transition Detection Latch

See [TDLA/B - Transition Detect Latches](#).

37.15.5.1.1.7 TCCEA - Transition Continuous Capture Enable

See [TCCEA - Transition Continuous Capture Enable](#).

37.15.5.1.2 Pin Control Registers

Pin Control Registers are replicated one per channel, accessed only by microcode and qualified by the CHAN register in the same way as Event Registers.

The following table lists Pin Control Registers, explained in following subsections, and their accesses.

Table 37-340. Pin Control Registers microcode accesses

Register	Access type	Sampled from channel	Update to channel	Microcode fields ¹	Reset value
IPACA, IPACB	write only	no	immediate	IPACA, IPACB	000,000
OPACA, OPACB	write only	no	immediate	OPACA, OPACB	000,000
PSTI	flag test on branch	no	no	BCC	0

Table continues on the next page...

Table 37-340. Pin Control Registers microcode accesses (continued)

Register	Access type	Sampled from channel	Update to channel	Microcode fields ¹	Reset value
PSTO	flag test on branch, write	no	immediate	BCC (test) PSC, PSCS (set)	0
OBE ²	write only	no	immediate	TBSA values 1000,1001	0 (negated)
PSS	flag test on branch	on CHAN assignment	no	BCC	n.a. ³
PRSS	flag test on branch	on CHAN assignment	no	BCC	n.a. ⁴

1. See [Microinstruction Set](#).

2. Although this register exists for code compatibility reasons, OBE functionality is not available on this chip.

3. PSS is PSTI or PSTO sampled on CHAN assignments and at thread start.

4. PRSS is PSTI sampled on channel service request.

37.15.5.1.2.1 IPACA/B and OPACA/B - Input and Output Pin Action Control Registers

These registers determine the transition detections and output pin actions which takes place due to match or transition events. Each field is three bit wide. After reset, the IPACA/B and OPACA/B registers are set to 000. IPACA and IPACB registers are mutually independent and have identical encoding, and so do OPACA and OPACB. The table below shows IPAC and OPAC encoding. Note that transition detections are independent from the output actions, but some options of output actions are triggered by transition detections. Output actions can also be triggered by Matches. For a detailed definition of Transition Detections, see [Transition detection and Time Base Capture](#).

IPACA/B define the Transition events treated by channel logic. Although the name "transition" is generically used for the transition detections, IPAC options 100 and 101 do not really detect transitions: they actually sample the state of the input signal at the occurrence of the corresponding Match (Match A used for IPACA, Match B used for IPACB).

Table 37-341. IPACA/B and OPACA/B Encoding

Value	IPAC meaning	OPAC meaning
000	Do not detect transitions	Do not change output signal
001	Detect rising edge only	Match ¹ sets output signal high
010	Detect falling edge only	Match ¹ sets output signal low
011	Detect either edge	Match ¹ toggles output signal
100	Detect input signal = 0 on Match ¹	Transition sets output signal low
101	Detect input signal = 1 on Match ¹	Transition detection sets output signal high

Table continues on the next page...

Table 37-341. IPACA/B and OPACA/B Encoding (continued)

Value	IPAC meaning	OPAC meaning
110	reserved	Transition detection toggles output signal
111	n.a. ²	n.a. ²

1. Match A is used for IPACA/OPACA, and Match B for IPACB/OPACB.
2. On the microinstruction fields IPACA/B and OPACA/B this value is neutral, meaning that IPAC/OPAC register values are not changed.

37.15.5.1.2.2 Output Pin Control Logic and Pin State Output Register - PSTO

The output signal control logic uses OPACA/B, the Pre-defined Channel Mode (PDCM) and the User Defined Channel Mode (UDCM), and the microcode Pin State Control (PSC and PSCS) fields. It is responsible for setting the Pin State Output (PSTO) register to the specified logic value required by microcode, by input events, or by Match A and/or Match B events. The PSTO register stores the driven pin state determined by the Pin Control logic. The Output Buffer Enable signal, if used at MCU integration, must be set by microcode (using TBSA field) to make the pad propagate the PSTO register output to the actual pin.

PSTO register output also goes to the microengine branch logic, enabling branching on the driven pin state. PSTO is set to 0 on reset.

The PSC and PSCS microcode fields are used for setting the PSTO register to a fixed value, or to the value specified by the OPACA or OPACB microcode field, as shown in the following table.

Table 37-342. PSC and PSCS encoding

PSC	Output pin action
00	Force pin state according to OPACA (PSCS=0) or OPACB (PSCS=1).
01	Force pin high.
10	Force pin low.
11	Do not change pin state.

See [Channel Control and Configuration Microoperations](#) for details.

On occurrence of match recognitions or transition detections, the pin control logic sets PSTO value according to the event number (Match A/Transition A or Match B/Transition B) and the contents of OPACA/IPACA or OPACB/IPACB registers. There are cases in which two match or transition events may occur at the same time, each of them trying to force a different pin action. The resolution of the selected match event which sets the

value depends on the Pre-defined Channel Mode (PDCM) register and the User Defined Channel Mode (UDCM). For details refer to [Match/Transition Pin Action Conflict Resolution](#).

37.15.5.1.2.3 PSTI and PSS - Pin State Input and Pin Sampled State Registers

During the time slot transition period, or whenever the CHAN register is written by microcode, the filtered²² input signal PSTI or output signal PSTO (selected by the ETPD field in [eTPU Channel n Configuration Register for Engine 1](#) (eTPU_CCCR_CnCR_ENG1) and [eTPU Channel n Configuration Register for Engine 2](#) (eTPU_CCCR_CnCR_ENG2)) is sampled into the branch logic as the PSS flag (see [Table 37-411](#)). The microcode can then branch on either the currently driven (PSTO) or input (PSTI) pin state, or on sampled pin state (PSS, which is stable as long as CHAN does not change).

Note

If a transition occurs simultaneously (after the filter) with the CHAN assignment, PSS samples the new pin value. Therefore, if TDLA/B is cleared simultaneously with the assignment CHAN := CHAN, the occurrence of a transition at this very moment can still be tested with PSS.

37.15.5.1.2.4 PRSS - Pin Request Service Sample

Channel logic can, depending on its state and programmed mode, request service to the eTPU microengine (see [Functions and threads](#)). When the channel logic issues a service request, the filtered input signal PSTI is sampled into an internal channel flag PRSS. There is one such PRSS flag for each channel (see [Pin Control Registers](#)). Channel PRSS keeps its value until all its service request sources are cleared and until a new channel service request rises.

NOTE

Only channel service requests update PRSS. It is not updated by host service requests (HSRs) or by link service requests. Moreover, a channel service request updates PRSS even if the channel is disabled (its CPR set to 0, see [eTPU Channel n Configuration Register for Engine 1](#) (eTPU_CCCR_CnCR_ENG1) and [eTPU Channel n Configuration Register for Engine 2](#) (eTPU_CCCR_CnCR_ENG2)).

22. The filter can be bypassed.

Functional description

The channel PRSS flag is sampled into the branch logic as the PRSS flag (see [Table 37-411](#)) during the time slot transition period, or whenever the CHAN register is written by microcode.

37.15.5.1.2.5 OBE - Output Buffer Enable control latch

NOTE

Output Buffer Enable is not available on this chip.

OBE latch drives the Output Buffer Enable signal, which can be used to control the output signal pad driver. Channel output comes disabled from reset. If an Output Buffer Enable signal²³ from eTPU is used on MCU integration and a signal is desired to be output in a channel, OBE signal must be set by microcode. Microcode field TBSA is used to set/reset the Output Buffer Enable control when microcode field TBSA bit 3 is 1, according to the following table.

Table 37-343. TBSA Output Buffer control

microcode TBSA[2:0]	meaning when TBSA[3]=1
000	enable output buffer
001	disable output buffer
111	do nothing
other values	reserved

23. There is one independent OBE signal for each channel.

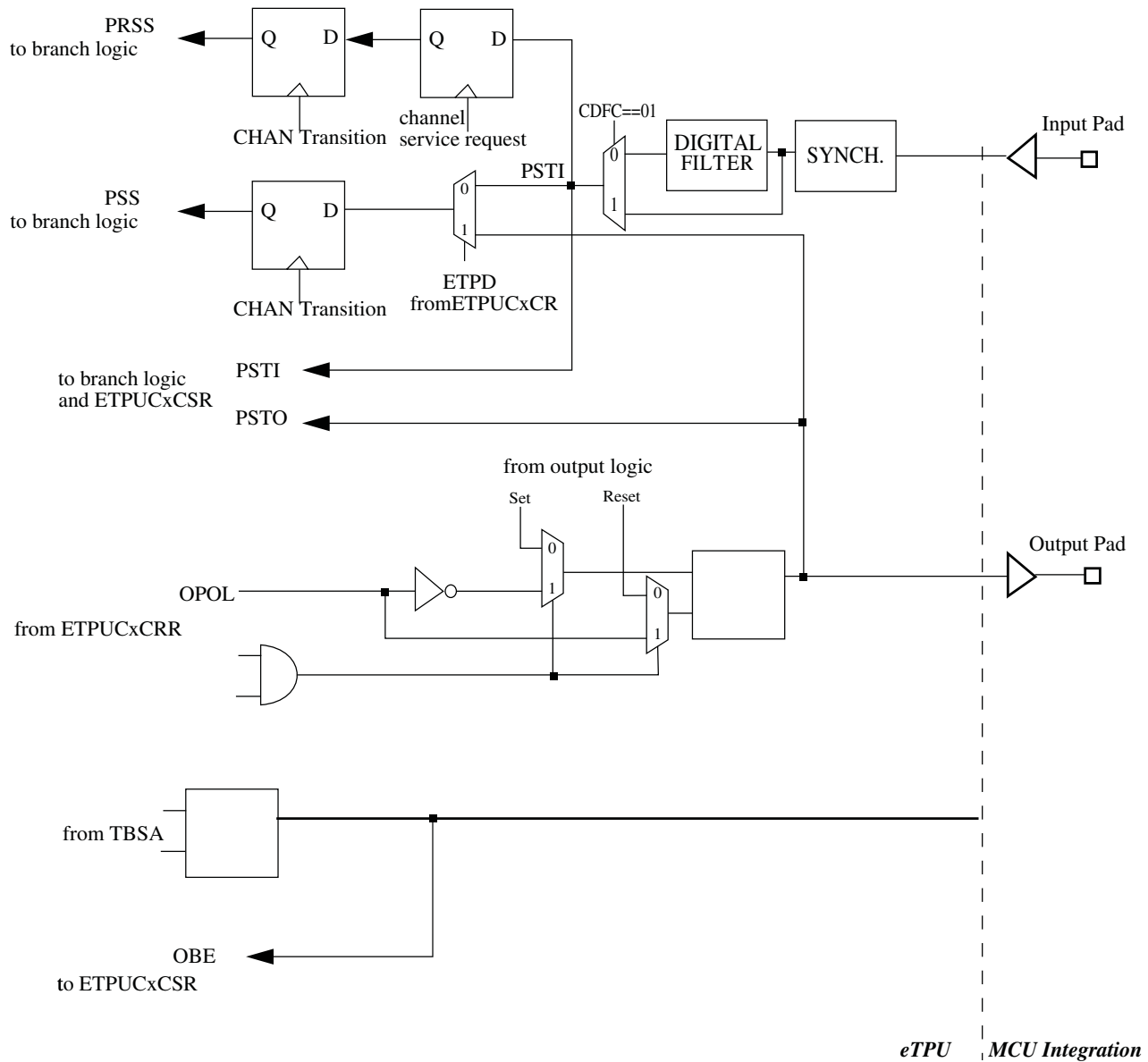


Figure 37-305. Pin State Input/Output Logic

37.15.5.1.3 General Channel registers

These registers control other aspects of channel logic. Except for CHAN, they are unique per channel. The following table summarizes the registers and access options.

Table 37-344. General Channel registers microcode access

Register	Access type	Sampled from channel	Update to channel	Microcode fields ¹	Reset value
CHAN	read/write	n.a. ²	n.a. ²	T4ABS, T4BBS, T2ABD	defaults to serviced channel at thread start

Table continues on the next page...

Table 37-344. General Channel registers microcode access (continued)

Register	Access type	Sampled from channel	Update to channel	Microcode fields ¹	Reset value
PDCM	write only	no	immediate	PDCM	1100 (sm_st)
UDCM	write only	no	from ERTA by microcode	CMW, ERWA	parameter value defined at integration
SRI	write only	no	immediate	MTD	1
Flag1, Flag0	branch flag test, write	no	immediate	BCC, FLC	0, 0

1. See [Microinstruction Set](#).

2. CHAN is common to all channels in the Engine.

37.15.5.1.3.1 Channel Selection Register - CHAN

CHAN is the register that holds the number of the channel that qualifies the context of most Channel Registers, including Pin Control and ER accesses, and is common to all channels in a same Engine.

When a thread starts to be executed, the contents of CHAN register is automatically updated on Time Slot transition to the number of the channel to be serviced. Serviced channel is constant during channel servicing, but the selected channel can be changed any time by microengine writing into CHAN register.

Some microinstructions are affected by the serviced channel instead of CHAN. These are:

- Conditional branch using LSR (see [LINK Register](#)) or Function Mode (see [eTPU Channel n Status Control Register for Engine 1 \(eTPU_CCCR_CnSCR_ENG1\)](#) and [eTPU Channel n Status Control Register for Engine 2 \(eTPU_CCCR_CnSCR_ENG2\)](#)).
- Negate channel flag LSR, Interrupt CPU and Data Transfer Request (see [Channel Interrupt and Data Transfer Requests](#)).

When CHAN register is written, accesses are qualified by the new CHAN register value from the instruction following CHAN assignment on, except CaptureA/B sampling into ERTA/B and Match register writing from ERTA/B (see [CHAN assignment, Read Match and ERWA/B](#)).

Writing CHAN (including with the same value, CHAN:= CHAN) updates ERTA and ERTB with the new captured values, the branch logic with updated MRLA/B and TDLA/B flags.

The following table shows the commands, flags and registers selected by the CHAN register value.

Table 37-345. CHAN-selected features

Feature used	Selected by CHAN
Channel-relative SDM access	YES
Branch using PSS, PRSS, PSTI and PSTO channel flags	YES
Branch using MRLA/B, TDLA/B, Flag0/1 ¹	YES
Branch on all other conditions ²	no
ERTA/B Value	YES
Configure (selected) channel	YES
Channel commands applied to: MRLA/B, TDLA/B, TBASA/B, IPACA/B, OPACA/B, PSC, PSCS, OBE, PDCM, MRLE, Flag0/1	YES
Channel command: set/reset SRI	YES
Channel command: write to Match registers (ERWA/B)	YES ³
Channel command: read Match registers into ERTA/B	YES
Clear LSR	no
Channel interrupts and data transfer requests (CIRC)	optionally

1. In TPU, these conditions retained the old values.

2. See [Branch Operations](#)

3. If write Match (ERWA/B) occurs at the same time of a CHAN assignment, the channel which is target of the write is the one selected by the new CHAN value. See [CHAN assignment](#), [Read Match](#) and [ERWA/B](#).

37.15.5.1.3.2 PDCM - Predefined Channel Mode

PDCM determines the channel mode assigned to the channel. Channel mode defines much of the channel logic behavior, specially how matches blocks and enables transitions and vice-versa, as well as occurrence of time base captures and service requests based on matches and transitions. For a complete description see [Channel Modes](#).

PDCM is a 4-bit register set by the microcode field of the same name (see [Predefined Channel Modes](#)), and cannot be read or tested in branch instructions. The table below relates the PDCM value with channel modes. The second column specifies the mnemonic used to reference the mode, introduced in [Channel Modes Overview](#). There is one PDCM for each channel, initialized with 1100 on reset.

PDCM is also used to select the User Programmable Channel Mode. If this selection is made, the channel behavior is defined by the settings of the UDCM register (see [UDCM - User Defined Channel Mode](#)).

Table 37-346. PDCM encoding

PDCM	Channel mode
0000	em_b_st

Table continues on the next page...

Table 37-346. PDCM encoding (continued)

PDCM	Channel mode
0001	em_b_dt
0010	em_nb_st
0011	em_nb_dt
0100	m2_st
0101	m2_dt
0110	bm_st
0111	bm_dt
1000	m2_o_st
1001	m2_o_dt
1010	User Defined Channel Mode
1011	reserved
1100	sm_st ¹
1101	sm_dt
1110	sm_st_e
1111	n.a. ²

1. This is the reset value, also compatible with TPU channel behavior.

2. This value is used as a neutral (do not change) value in the PDCM microinstruction field.

37.15.5.1.3.3 UDCM - User Defined Channel Mode

UDCM holds the control signals that define the channel logic behavior in terms of Match and Transition blocking and enabling, captures and service requests triggered by events. The register layout is defined in the figure below. The individual fields are defined in [Channel Mode Logic and Event Flags](#). There is one UDCM register for each channel, which can be independently programmed.

UDCM can only be written into the Match register by microcode, through ERTA/B microengine registers (see [Write Channel Match and UDCM Registers](#)).

13	12	11	10	9	8	7	6	5	4	3	2	1	0
MSR	MCAP	M1ET	M1EM2	M1BM2	M2BM1	M2BT	T1BM1	T2BM1	TBM2	T1ET2	TSR	TCAP	

Figure 37-306. UDCM Register

37.15.5.1.3.4 SRI - Match/Transition Service Request Inhibit Latch

SRI blocks channel service requests due to the assertion of MRLA/B and/or TDLA/B. SRI does not affect recognition of Link Service Requests or Host Service Requests, neither MRLA/B or TDLA/B microcode branch tests nor Entry Table selection²⁴. SRI is asserted during reset and is controlled by microcode field MTD.

24. In TPU, SRI also blocked TDL and MDL branches and enabled any transition to capture time base.

To unburden the microengine, SRI asserted configures a channel "dumb" regarding the servicing of match and capture channel service requests. Even with SRI=1, TDLA/B and MRLA/B can still be asserted, and the level specified by the OPAC (Output Pin Action Control) registers will be output to the pin.

37.15.5.1.3.5 Flag1,Flag0 - Channel "state resolution" flags

Each channel has a pair of flags, simply called Flag0 and Flag1, that can be set/reset by microcode through microinstruction field FLC. FLC sets/resets Flag0/1 of the channel selected by CHAN. These flags can be tested by microcode, and are also used to resolve the microcode entry point for the channel service (see [Entry Points](#)). Flag0 and Flag1 are, so, typically used for fast state resolution. FLC microinstruction field also allows Flag1,Flag0 to be copied from selected bits of P register high byte, which is also meant to be used to hold application state. Flag0 and Flag1 are both zero out of reset.

37.15.5.2 Match Recognition

The match operation is performed every microcycle by comparing the channel MatchA and MatchB registers against the value of the TCR bus specified for each match. TCR1 or TCR2 bus is selected according to TBSA and TBSB fields. Both results have effect on the next clock cycle (see [Microcycle and I/O timing](#)).

A Match A/B event is qualified by a set of match enabling conditions to the Match Recognition Registers MRLA/B. To recognize the match, the following match enabling conditions are required:

- For IPACA/B=0xx, Match Enable Flag (MEF), qualified by the channel currently being serviced must be asserted. For IPACA/B =1xx, Match A/B is always enabled (even during Time Slot Transition - TST), regardless the state of the Match Enable Flag (MEF). See [MEF - Match Enable Flag](#) for the conditions of MEF assertion.
- Match Recognition Latch Enable 1/2 (MRLEA/B) is asserted. A match event recognition may only occur if its corresponding MRLEA/B bit is set, which only happens upon a write to a channel match register by the microcode, copied from ERTA/B. MRLEA/B is negated when the respective match occurs or, in some double match channel modes, when a match for the other Match register occurs. It ensures that the greater-equal comparison will not cause additional matches²⁵.
- In selected modes (see [Channel Modes](#)), the particular conditions of MRL and TDL flags of the other event, i.e:

25. Microcode can also negate MRLEA/B.

- MRLA, TDLA enable or block MRLB;
- MRLB, TDLB enable or block MRLA.
- The respective MRL is negated.
- In selected modes (see [Channel Modes](#)), the state of its respective TDL flag.

If the Match A and/or Match B conditions are met, the channel immediately forces the pin state if specified by the appropriate OPACA/B registers (Output Pin Action Control 1/2) and, in some cases, by IPACA/B registers. Refer to [IPACA/B and OPACA/B - Input and Output Pin Action Control Registers](#).

If both Match A and Match B events occur at the same time, with conflicting pin actions, the priority over the pin action is mode dependent. For further details on pin action resolution refer to [Match/Transition Pin Action Conflict Resolution](#).

37.15.5.2.1 MRLA/B - Match Recognition Latches

MRLA/B indicate the recognition of a match event detected by the comparator. They can be asserted either on T2 or T4 (see [Microcycle and I/O timing](#)). Assertion of MRLA/B issues a match Service Request in specific channel modes, depending on previous events and state of SRI. After reset MRLA and MRLB are both negated.

When MRLA or MRLB is asserted, it may change the output signal level according to the Input and Output Pin Action Control registers (refer to [IPACA/B and OPACA/B - Input and Output Pin Action Control Registers](#)). Assertion of MRLA/B causes a capture of one or two time bases, according to the selected mode capturing scheme (see [Transition detection and Time Base Capture](#))

A match recognition is self-blocking, regardless of Channel Mode: once MRLA (MRLB) has been asserted, it negates its associated MRLEA (MRLEB) register, preventing future match recognitions, until the associated match register is rewritten by microcode. The microcode has to enable new matches by updating the new match value in the MatchA (MatchB) register²⁶. In addition, assertion of MRLA/B can block its twin MRLB/A, depending on the channel mode. In some double match blocking channel modes, Match A/B event blocks the occurrence of Match B/A in a "first win" scheme.

It is the transition from 0 to 1 in MRL that causes the Match actions, except the service request: apart from MRLEA/B negation(s) and service request, no action due to a Match occurs if MRL was already set to 1, even if the other MRL assert conditions are satisfied. However, if a Match and a microoperation negating its corresponding MRL occur at the

26. Before that, microcode should also negate MRLA (MRLB), otherwise an old match may be recognized by the scheduler and serviced as a new one

same time, MRL negation by microcode overrides its assertion, but any dependable captures and pin action occurs anyway (if MRL was already negated before), and also the negation of MRLE(s) (the respective one and, in some channel modes, the other, regardless of MRL state before). Note that MRLE must have been set before (by writing a new Match value).

37.15.5.2.2 MEF - Match Enable Flag

MEF is a one-bit latch that is unique for all channels in an Engine.

MEF can selectively enable assertion of MRLA/B, depending on the IPACA/B field. For IPACA/B=0xx, MEF=1 enables assertion of MRLA/B for the scheduled channel during service. For IPACA/B=1xx, Match A/B is always enabled, regardless the state of the MEF, but it still depends on the other Match recognition conditions. Matches of channels not being serviced are never disabled by MEF.

MEF is not accessible by Microengine or Host. MEF is negated for two microcycles in the beginning of the time slot transition period. After two microcycles (plus wait-states) into TST, the ME bit in the entry point is copied to MEF to allow selective enabling of MRL for each thread (refer to [Time Slot Transition](#)). MEF is asserted unconditionally soon after a thread ends.

If a channel service needs to postpone a programmed match, MEF assures that microcode wins the race against match event after time slot transition (only for IPAC=0xx).

Note that a match event may be lost during the periods when MEF is negated only if:

- the match comparator is configured for "equal-only" behavior, and
- IPACA/B=0xx, and
- TCR increments at the rate of eTPU clock divided by 2 or faster.

When the comparator is configured as "greater-equal", the match event that occurred when MEF was negated may be recognized after MEF is asserted again, due to the "greater than" condition.

37.15.5.2.3 MRLEA/B - Match Recognition Latch Enable

MRLEA/B is negated upon the assertion of its respective MRLA/B. In blocking match channel modes it may also be negated together with the assertion of the twin MRLB/A. The MRLEA/B bits ensure that data captured due to the first match event will not be overwritten when MRLA/B is negated: due to greater-equal comparison, the match condition continues to be true, but should not cause another capture event.

In addition to negation by local match event, the microcode can negate both MRLEA and MRLEB, to block pending matches, and also MRLA/B, individually. This action will prevent future match events from the selected channel.

Writing the MatchA/B registers by microcode to schedule the next match values sets MRLEA and/or MRLEB and enables new matches. This setting overrides the MRLE negation conditions due to channel logic or microcode (see [Channel Modes](#)). By combining write to Match A/B with MRLEA/B negation microinstructions, the microcode can negate one MRLE while asserting the other.

Note

If the MRLE negation conditions continue after writing MatchA/B registers, the respective MRLE does not keep asserted. For instance, if MRL=1 and a match is programmed for a time value in the past during a thread with ME=1, then MRLE will be cleared soon after MatchA/B is written, even though a match does not occur (because MRL was already asserted, neither captures nor pin toggles occur).

When the match register is updated (with MRLE already asserted before) and field MRLA/B = 1 (no clear, see [Clear Transition/Match Event Registers](#)) and MRLA/B flag is zero, the eTPU behaves exactly as the TPU, that is: a match that comes concurrently with the rewrite of the match register, matching the old value, sets the MRL, as if the setting of the MRLE due to match register write had precedence over its clear by the match at that moment. After this simultaneous operation, the MRLE value stays at 1, and the captured time base value, if any, reflects the match value.

When the match register is updated (with MRLE already asserted before) and field MRLA/B = 0 (clear MRL, see [Clear Transition/Match Event Registers](#)) and MRLA/B flag is zero, the match captures will occur, the MRLA/B flag will keep negated, and MRLE will stay asserted. Special care must be taken when a match is reprogrammed on a channel running at T2/T4 timing, please see the details in the “T2/T4 Channel Timing” section.

37.15.5.3 Transition detection and Time Base Capture

Time Base Capture(s) occur when the value of a specified TCR is sampled into the CaptureA and/or CaptureB register. TBSA[1] and TBSB[1] select which TCR will be captured in CaptureA and CaptureB, respectively.

A capture event may occur due to either of the following events:

- The assertion condition of Match Recognition Latch (MRL), even if MRL is simultaneously negated by microcode
- The assertion condition of Transition Detection Latch (TDL), even if TDL is simultaneously negated by microcode.
- Any Transition Event specified by IPACA if both the Transition Detection Latch TDLA and Transition Continuous Capture Enable TCCEA are asserted.

A capture event occurs together with the assertion of MRL or TDL either on T2 or T4 positive edges, and captures the time-base value that caused the match, even if TCR1/2 increments concurrently with the assertion (see [Microcycle and I/O timing](#))²⁷. MRLA/B and TDLA may, depending on the channel mode, inhibit the capture of the second event's TCR into CaptureA/B. As a general rule, values captured by signal transitions are not overwritten by values captured by match events.

When the enable bit TCCEA is asserted, captures due to Transition Events also occur after TDLA is asserted. Those captures happen on transition events specified by IPACA, and the TCR value is saved into CaptureA register only.

The capturing scheme is defined by the Channel Mode programmed at register PDCM, or at register UDCM when User Defined Channel Mode is selected. For more information on mode-dependent capture schemes refer to [Channel Modes](#).

37.15.5.3.1 TDLA/B - Transition Detect Latches

TDLA/B indicate detection of specific transition occurrences on a channel input signal. TDLA and TDLB assertion causes service request in single and double transition predefined modes, respectively. TDLB assertion does not cause Service Request in single transition predefined modes, and TDLA assertion does not cause Service request in double transition predefined modes. In single transition channel predefined modes TDLB can be asserted on the second transition, but does not generate Service Request. Yet on predefined modes, TDLB assertion is enabled only if TDLA is asserted to detect an ordered input signal double transition. All the restrictions above, however, may be overridden by using the User Defined Channel Mode. The IPACA and IPACB registers indicate the programmed edges of the first and second detected transition, respectively.

The sampling of a determined value (0 or 1) on the input signal due the occurrence of a Match is also treated as a "transition", depending on IPACA/B programming (see [IPACA/B and OPACA/B - Input and Output Pin Action Control Registers](#)). When using a channel mode where the transition1 is initially blocked and IPACA is programmed to detect such "transitions", the occurrence of a Match A only unblocks the transition after

27. In TPU3, when TCR1 was counting at maximum rate of eTPU clock divided by 2, the next value was captured.

the sampling. It means that the transition on the first Match (IPACA configurations 100 and 101) is not effective on predefined modes where Transition A is enabled by Match A (m2_st, m2_dt, m2_o_st and m2_o_dt) or user-defined modes with UDCM bit M1ET asserted. A Transition A can still happen after the Match A, however, if MatchA register is reprogrammed without clearing MRLA.

TDLA/B assertion conditions initiates a capture event of one or both selected TCR buses. TDLA or TDLB transition event generates a Service Request, depending on channel mode, previous events and the state of SRI. For more information on service request scheme refer to [Entry Point address generation](#) and [Channel Modes](#).

Assertion of TDLA/B occurs on either T2 or T4 positive edges. The capture event occurs on the same clock, and captures the time base value present when TDLA/B was asserted²⁸. TDLA and TDLB are negated during reset and may also be negated independently by microcode. TDLA/B is reset by no way other than reset and microcode.

It is the transition from 0 to 1 in TDL that causes the Transition actions: even if TDL assert conditions are satisfied, no action due to a Transition occurs if TDL was already set to 1. However, if a Transition and a microoperation negating TDLs occur at the same time and TDL was already negated, TDL negation by microcode overrides its assertion, but any dependable captures and pin action occurs anyway.

37.15.5.3.2 TCCEA - Transition Continuous Capture Enable

TCCEA enables capture from transitions after the TDLA flag is set. TCCEA is negated on reset, so that a capture occurs only when TDLA asserts. TCCEA can be set and reset by microcode only, through the instruction field MTD (see [Disable Match and Transition Service Requests](#)). It can only be set together with inhibiting of the channel service requests (SRI = 1)²⁹.

When TCCEA is asserted, the transition events specified by IPACA that occur after TDLA is set also cause captures, into the CaptureA register only. However, output actions related to transition events are still blocked by TDLA.

28. In TPU3, when TCR1 was counting at maximum rate of eTPU clock divided by 2, the next value was captured.

29. TCCEA provides compatibility with TPU when service request is inhibited.

37.15.5.4 Channel Modes

The Enhanced Channels support various modes of operation combining Match A/B recognition and transition detection events which set MRLA/B and TDLA/B. The channel mode is individually set for each channel by eTPU microcode, through the PDCM register (see [PDCM - Predefined Channel Mode](#)). The PDCM register selects among a set of 13 predefined channel modes, and also a user-defined channel mode.

The order in which events occur, combined with assigned channel mode, establish which following event detections are inhibited and/or enabled, as well as the actions taken: Time Base capture, flag setting (MRLA/B, TDLA/B), match disabling (MRLEA/B), output signal transition, and Service Request. Those channel mode characteristics are fixed in the predefined modes, but can be individually programmed in the user-defined channel mode.

A generic description of channel modes from the usage point of view can be found in [Channel Modes Overview](#). Each mode is named with a mnemonic acronym for terse reference. The individual programmed attributes of the user-programmable channel mode are also described.

The modes are used differently for input and output signals, as explained in [Predefined Channel Modes on Input Signal Processing](#) and [Channel Modes on Output Signal Generation](#). Modes also allow combining input processing and output generation in a single channel, as exemplified in [Combining Input and Output Signals](#).

[Channel Mode Logic and Event Flags](#) shows a structural definition of channel logic and its relation to channel modes. A dynamic, event-oriented view of each channel mode can be found in [Channel Mode Summary](#).

37.15.5.4.1 Channel Mode Logic and Event Flags

[Figure 37-307](#) shows a more detailed diagram of channel mode logic. The logic shown here is not necessarily identical to the actual channel logic implementation, but is equivalent with respect to conditions for event blocking, enabling, capture, and service requests.

Signals MSR, TSR, MCAP, TCAP, M1ET, M1EM2, M1BM2, M2BM1, M2BT, T1BM1, T2BM1, TBM2 and T1ET2 are decoded from programmed channel mode PDCM in predefined modes, and come directly from the UDCM register when user-defined mode is selected:

- TSR (1 bit) defines Service Requests issued by Transitions, as shown in [Table 37-348](#).
- MSR (2 bits) defines Service Requests issued by Matches, as shown in [Table 37-347](#).

- TCAP (1 bit) defines time base captures caused by Transitions, as shown in [Table 37-348](#)
- MCAP (1bit) defines time base captures caused by Matches, as shown in [Table 37-349](#).
- M1ET, M1EM2, M1BM2, M2BM1, M2BT, T1BM1, T2BM1, TBM2, T1ET2 (1 bit each) define Match and Transition reciprocal blocking and enabling, as well as Transition ordering, as shown in [Table 37-350](#) and [Table 37-351](#).

[Table 37-352](#) shows the decoded values of those control signals for each predefined channel mode. The first column shows the mnemonic reference for the predefined channel modes described in the following sections.

Changing PDCM or the UDCM when user mode is selected may set or reset any of the channel flags, or issue captures and service requests, so it is advisable to switch channel modes only in a quiescent channel state, with channel flags MRLEA/B, TDLA/B, MRLA/B cleared. Furthermore, an event flag asserted in one mode may keep asserted after the mode programming change, even if the flag is impossible to be set in the new mode.

Table 37-347. MSR signals - Match Service Requests

value	MSR
00	issue no Service Requests on Matches
01	issue Service Request on Match B only
10	issue Service Request on 2nd ¹ Match
11	issue Service Request on both Matches

1. 2nd Match means the Match that happens after the 1st Match in time, either Match A or Match B.

Table 37-348. TCAP and TSR signals - Transition Captures and Service Requests

value	TCAP	TSR
0	1st ¹ Transition captures Time Bases corresponding to Transition A and Transition B ² Transition A captures corresponding Time Base if it is the second transition; Transition B captures corresponding Time Base if it is the second transition.	issue Service Request on the 1st ¹ Transition
1	Transition A captures corresponding Time Base. Transition B captures corresponding Time Base.	issue Service Request on the 2nd ³ Transition

1. 1st Transition means the Transition that happens first in time, either Transition A or Transition B.
2. Match capture(s) never overwrites a Transition capture. Transition captures can always overwrite a Match capture.
3. 2nd Transition means the Transition that happens second in time, either Transition A or Transition B

Table 37-349. MCAP signal - Match Capture

value	MCAP
0	Match A captures corresponding Time Base; Match B captures corresponding Time Base
1	either Match captures Time Bases corresponding to Match A and Match B ¹

1. Match capture(s) never overwrites a Transition capture. Transition captures can always overwrite a Match capture.

Table 37-350. TBM2 signal - Transition Blocks Match B

value	TBM2
0	Transition A Blocks Match B
1	Transition B Blocks Match B

Table 37-351. M1ET, M1EM2, M1BM2, M2BM1, M2BT signals

signal	active value meaning
M1ET (Match A Enable Transitions)	Transitions are initially blocked, and Match A enables Transitions
M1EM2 (Match A Enables Match B)	Match B is initially blocked ¹ , and Match A enables Match B
M1BM2 ² (Match A Blocks Match B)	Match B is initially enabled ¹ , and Match A blocks Match B ³
M2BM1 ² (Match B Blocks Match A)	Match A is initially enabled, and Match B blocks Match A ³
M2BT (Match B Blocks Transitions)	Match B blocks Transitions
T1BM1 (Transition A Blocks Match A)	Transition A blocks Match A
T2BM1 (Transition B Blocks Match A)	Transition B blocks Match A
T1ET2 (Transition A Enables Transition B)	Transition B is initially blocked, and Transition A enables Transition B

1. The initial condition of M1EM2 prevails over M1BM2, while M1BM2 blocking prevails over M1EM2 enabling, so that Match B stays always blocked when both M1BM2 and M1EM2 are active. This combination is used in single-match modes (sm_*).
2. Blocking of one Match by the other is done through MRLEs.
3. Matches always block themselves by resetting their own MRLEs (Match A always blocks Match A, Match B always blocks Match B)

Table 37-352. Predefined Channel Mode control signals decoding

Predefined Mode	M SR	MCAP	M1 ET	M1EM2	M1BM2	M2BM1	M2 BT	T1BM1	T2BM1	TBM2 ¹	T1ET2	TS R ¹	TCAP ¹
em_nb_st	11	0	off	off	off	off	off	On	off	0	On	0	0

Table continues on the next page...

Table 37-352. Predefined Channel Mode control signals decoding (continued)

Predefined Mode	M SR	MCAP	M1 ET	M1EM2	M1BM2	M2BM1	M2 BT	T1BM1	T2BM1	TBM2 ¹	T1ET2	TS R ¹	TCAP ¹
em_nb_dt	11	0	off	off	off	off	off	On	off	1	On	1	1
em_b_st	11	1	off	off	On	On	off	On	off	0	On	0	0
em_b_dt	11	1	off	off	On	On	off	On	off	1	On	1	1
bm_st	10	0	off	off	off	off	off	On	off	0	On	0	0
bm_dt ²	10	0	off	off	off	off	off	off	On	1	On	1	1
m2_st	01	0	On	off	off	On	off	On	off	0	On	0	0
m2_dt	01	0	On	off	off	On	off	On	off	1	On	1	1
m2_o_st	01	0	On	On	off	off	On	On	off	0	On	0	0
m2_o_dt	01	0	On	On	off	off	On	On	off	1	On	1	1
sm_st	11	1	off	On	On	On	off	On	off	0	On	0	0
sm_dt ²	11	1	off	On	On	On	off	off	On	1	On	1	1
sm_st_e ³	11	0	off	On	On	On	off	On	off	0	On	0	0

1. signals TSR, TCAP and TBM2 replace the signal DTM used in previous eTPU versions.
2. bm_dt and sm_dt are exceptions in the match blocking logic by transitions. See [Both Match Request Modes \(bm_st, bm_dt\)](#) and [Single Match Modes \(sm_st, sm_dt\)](#).
3. sm_st_e is an exception in the capture scheme. See [Single Match Enhanced Mode \(sm_st_e\)](#).

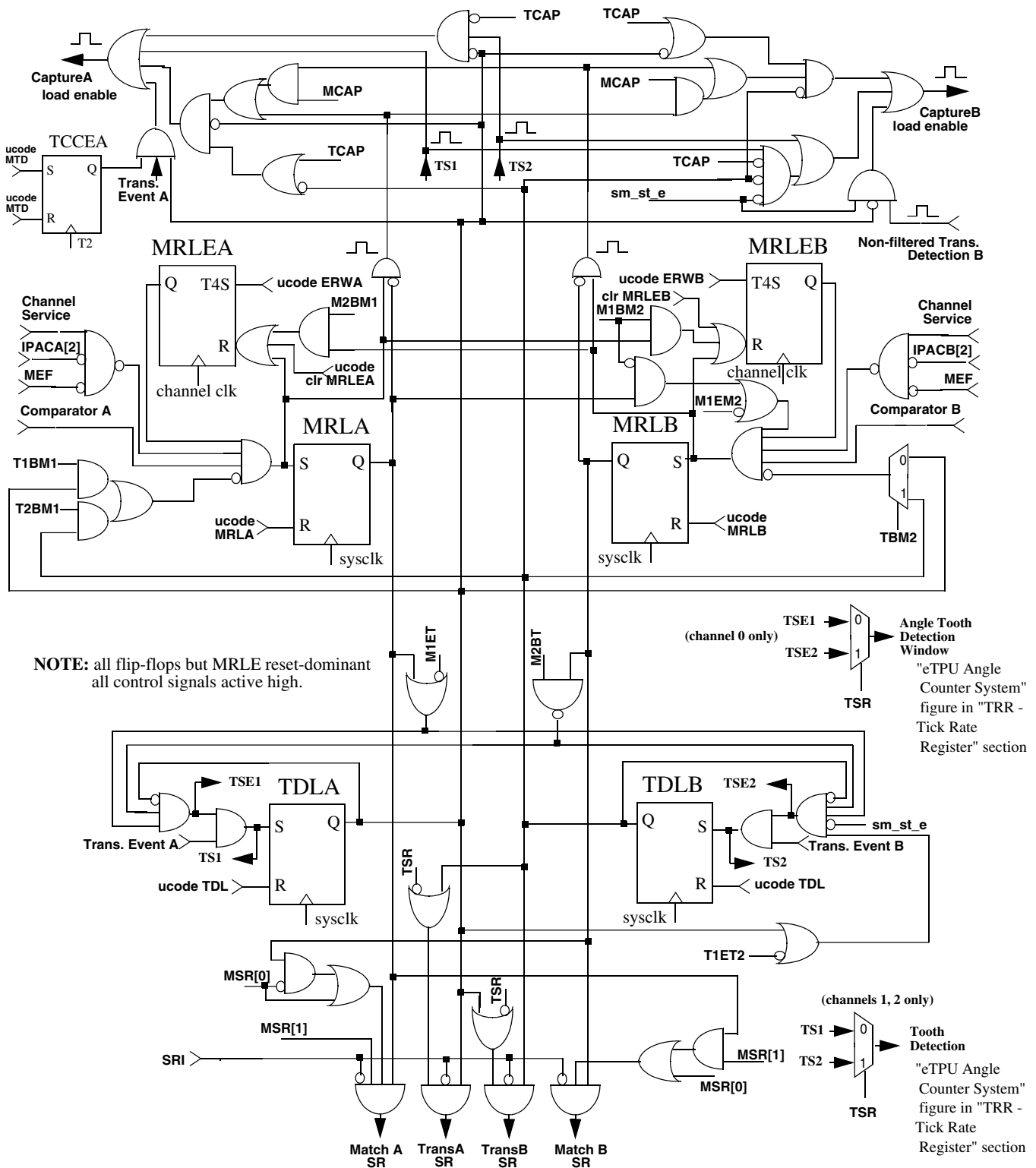


Figure 37-307. Channel Mode Logic and Event Flags

37.15.5.4.2 Channel Modes Overview

The predefined channel modes are divided, according to the way they treat transitions, in two basic modes:

- **Single Transition Modes** (mnemonic suffix **_st**): in these modes the first transition (flagged in TDLA) issues a service request, and captures both time bases (selected by TBSA[1] and TBSB[1]) except on sm_st_e. The second transition (flagged in TDLB) does not issue a service request, but it captures time base selected by TBSB[1], except on sm_st_e.
- **Double Transition Modes** (mnemonic suffix **_dt**): in these modes the second transition (flagged in TDLB) issues a service request, and each transition captures its own selected time base (Transition A and Transition B capture time bases selected by TBSA[1] and TBSB[1], respectively).

In predefined modes, Transition B is always (but not only) enabled by Transition A, so that transitions are always ordered: TDLA is set on the first transition and TDLB on the second. Unordered transitions are possible with user-defined mode, when UDCM bit T1ET2=0. Matches are generally not ordered, except on specific ordered match modes m2_o_st and m2_o_dt. Match capture(s) never overwrites a Transition capture, while Transition captures can always overwrite a Match capture, either in predefined or user-defined modes.

The following general rules apply to both predefined and user-defined modes:

- Blocking of one Match by the other, when it occurs, is done through MRLEs.
- Matches always block themselves by resetting their own MRLEs (Match A always blocks Match A, Match B always blocks Match B).

Predefined modes differ mostly by the way matches affects and are affected by other matches and transitions, as explained in next sections. However, some general rules on Match blocking apply:

- Match B is blocked by first transition (TDLA) in single transition modes, and by second transition (TDLB) in double transition modes.
- Both Matches are blocked by first transition in single transition modes.

Note

The rules above and in following sections may be overruled by the state of the channel latches if they are set/reset by microcode or if channel mode is changed. Care must be taken

to change channel modes, and is advisable to reset channel flags MRLA/B, TDLA/B and MRLEA/B before writing PDCM, or to UDCM when user-defined mode is selected.

37.15.5.4.2.1 Either Match, Blocking Modes (em_b_st, em_b_dt)

In these modes the first match recognition that occurs blocks the other match recognition and generates a service request. They end up with one service request for two programmed match recognitions where only the first match recognition has an actual effect. If both match recognitions occur at the same time, both MRLA and MRLB are set, before the mutual blocking takes effect. The associated logic is shown in the following figure.

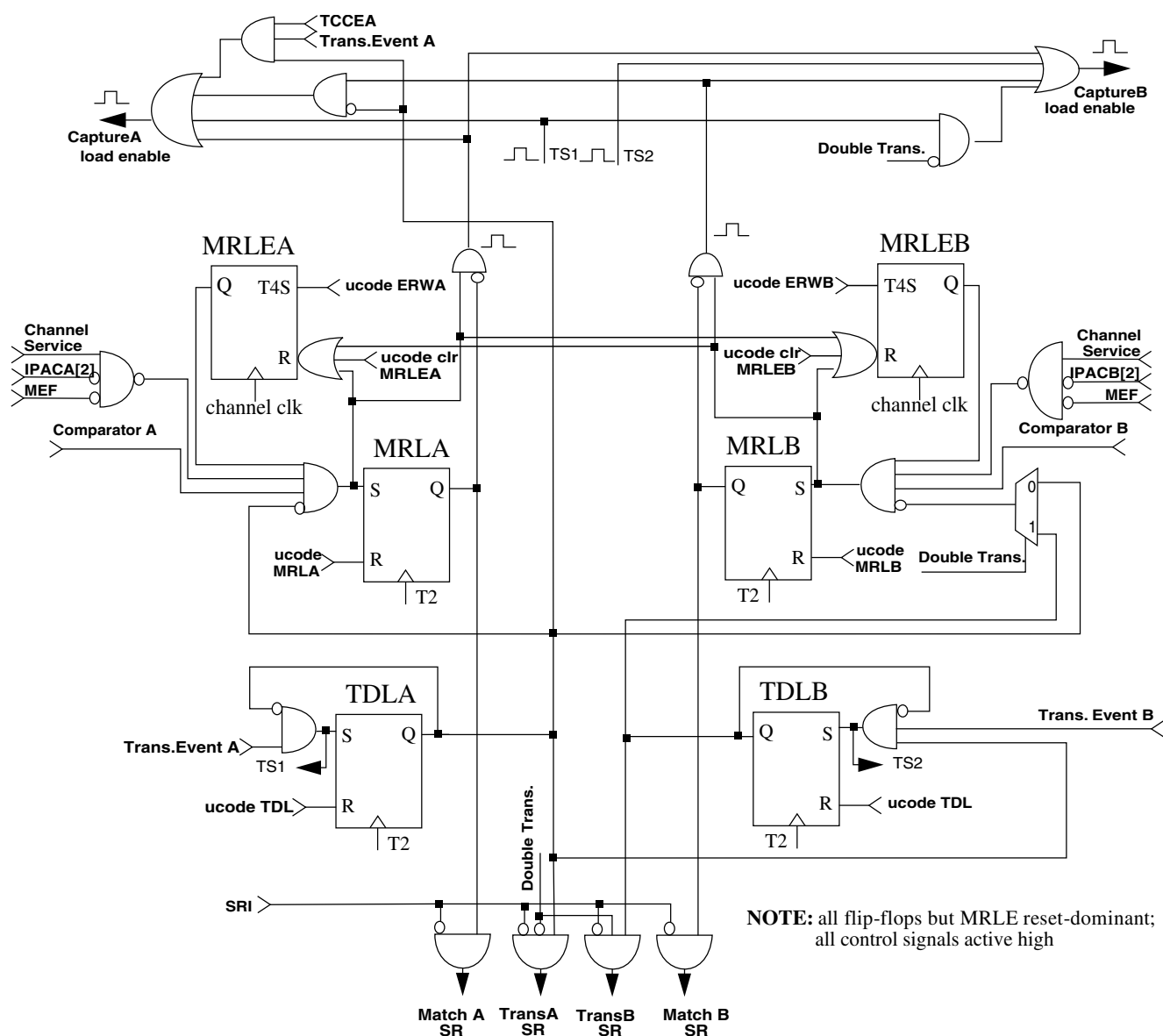


Figure 37-308. Either Match, Blocking Modes (em_b_st, em_b_dt)

37.15.5.4.2.2 Either Match, Non Blocking Modes (em_nb_st, em_nb_dt)

In these modes both match recognitions are independent and each of them generates service request. Each match recognition captures its related time base and does not block the other. The associated logic is shown in the following figure.

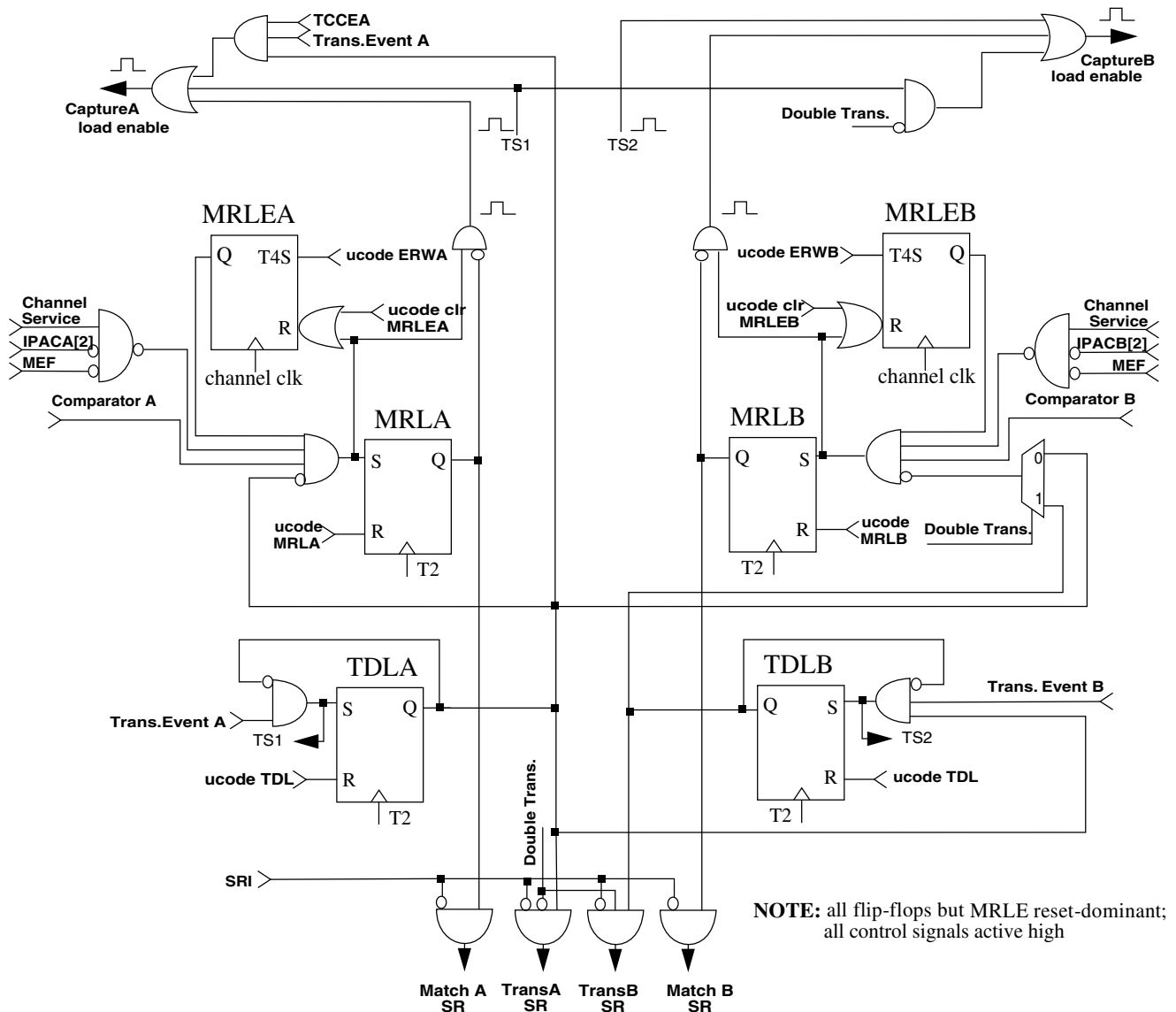


Figure 37-309. Either Match, Non Blocking Modes (em_nb_st, em_nb_dt)

37.15.5.4.2.3 Match B Request Modes (m2_st, m2_dt)

In these modes transitions are initially blocked, and are enabled by Match A. Match B recognition generates the match service request and disables Match A recognition. Each match recognition captures its own programmed timebase. In case of simultaneous match

recognition, both MRLA and MRLB are set, and OPACB register has priority over OPACA for selecting the pin action. The associated logic is shown in the following figure.

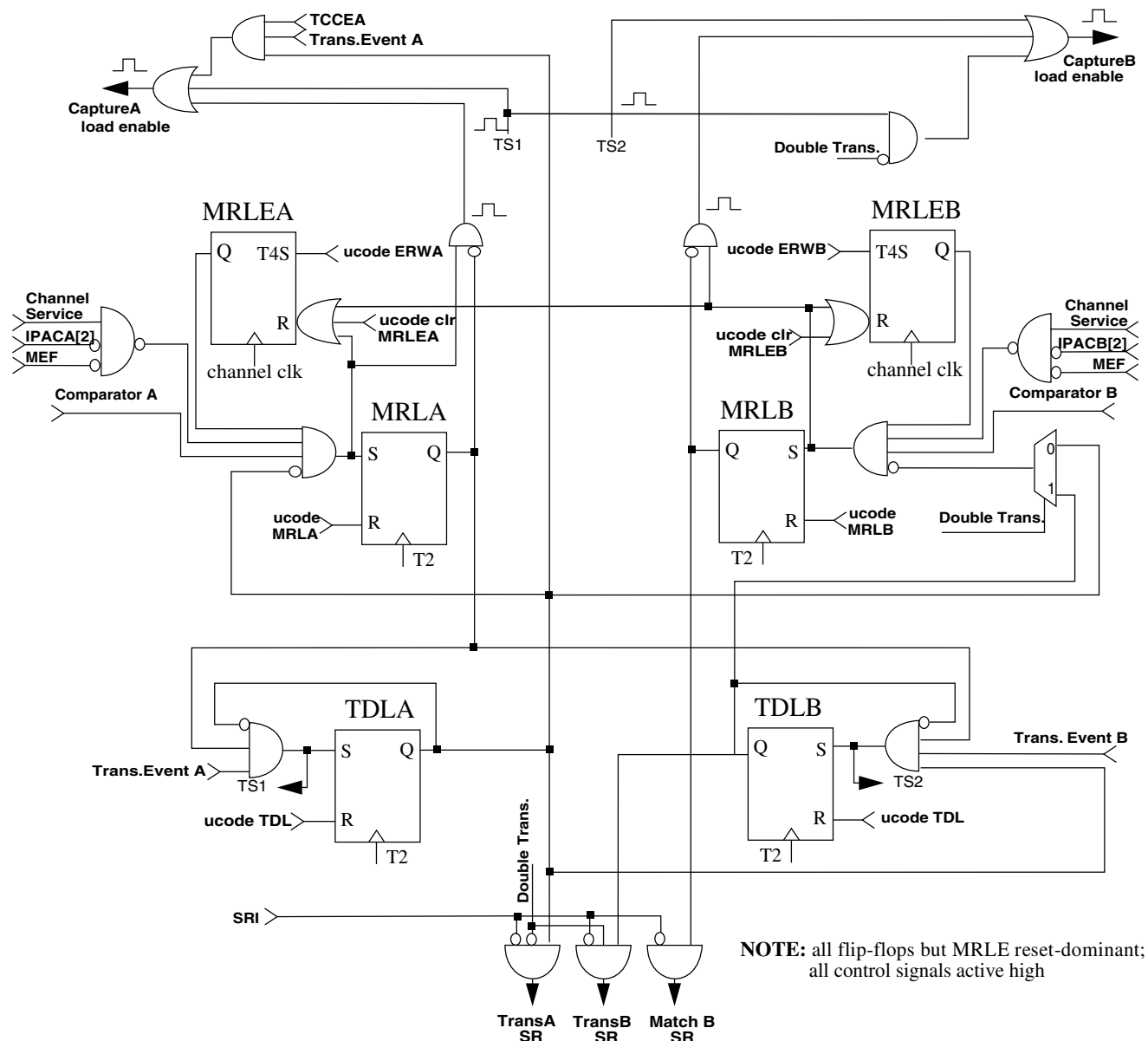
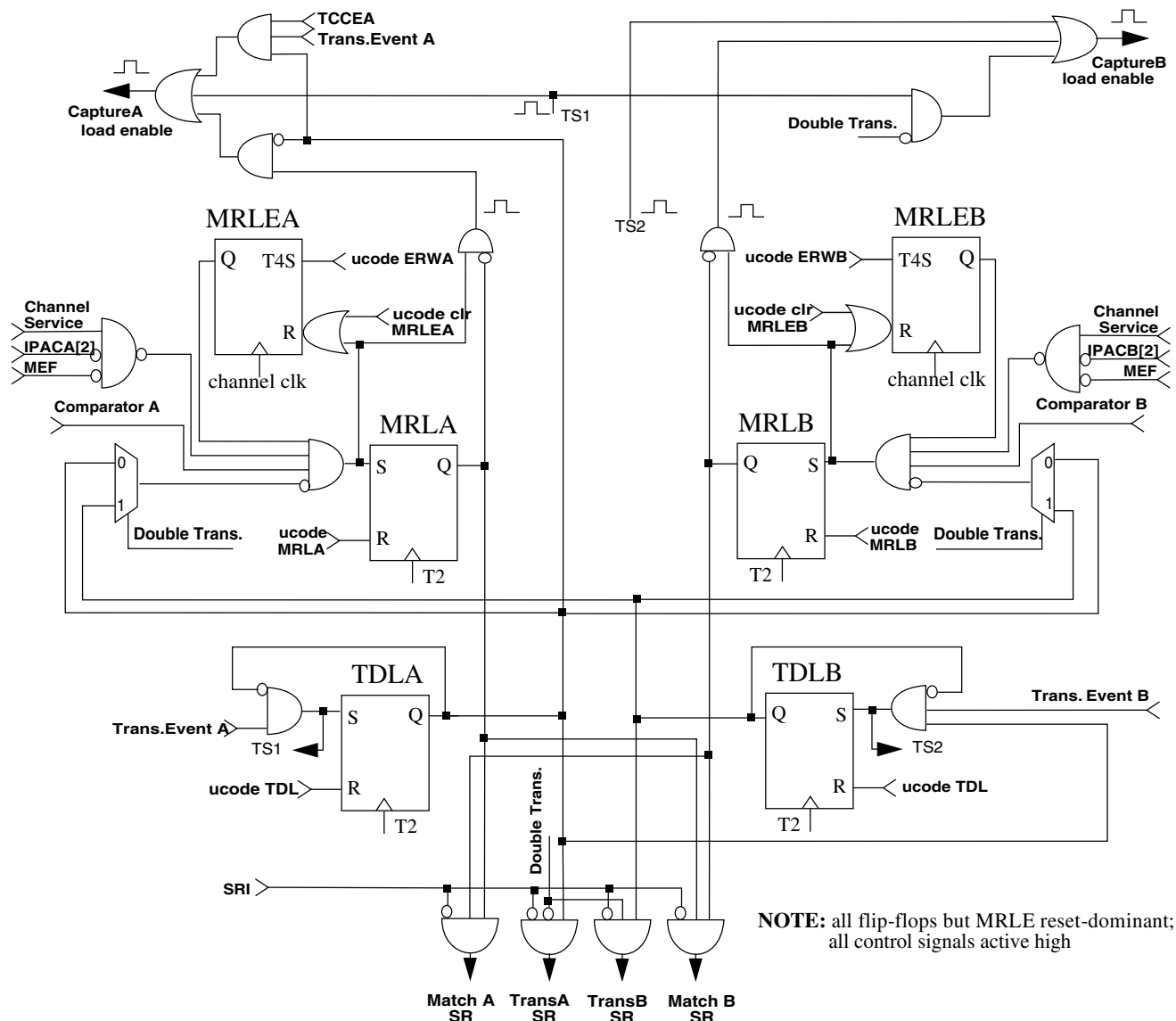


Figure 37-310. Match B Request Modes (m2_st, m2_dt)

37.15.5.4.2.4 Both Match Request Modes (bm_st, bm_dt)

In these modes, match service request is generated only after both match recognitions occurred. By definition this is a non-blocking match mode: match recognitions do not block each other, implementing a last-served scheme. Unlike other double transition

modes, bm_dt blocks Match A with Transition B (not with Transition A), so that the second transition blocks both matches. The associated logic is shown in the following figure.



37.15.5.4.2.5 Ordered Modes with Match B Request (m2_o_st, m2_o_dt)

These are ordered match modes on which Match A recognition must precede Match B recognition (ordered 1->2). Match A asserts MRLA and enables Match B and transitions. Match B asserts MRLB, generates a match service request, and blocks both transitions. The associated logic is shown in the following figure.

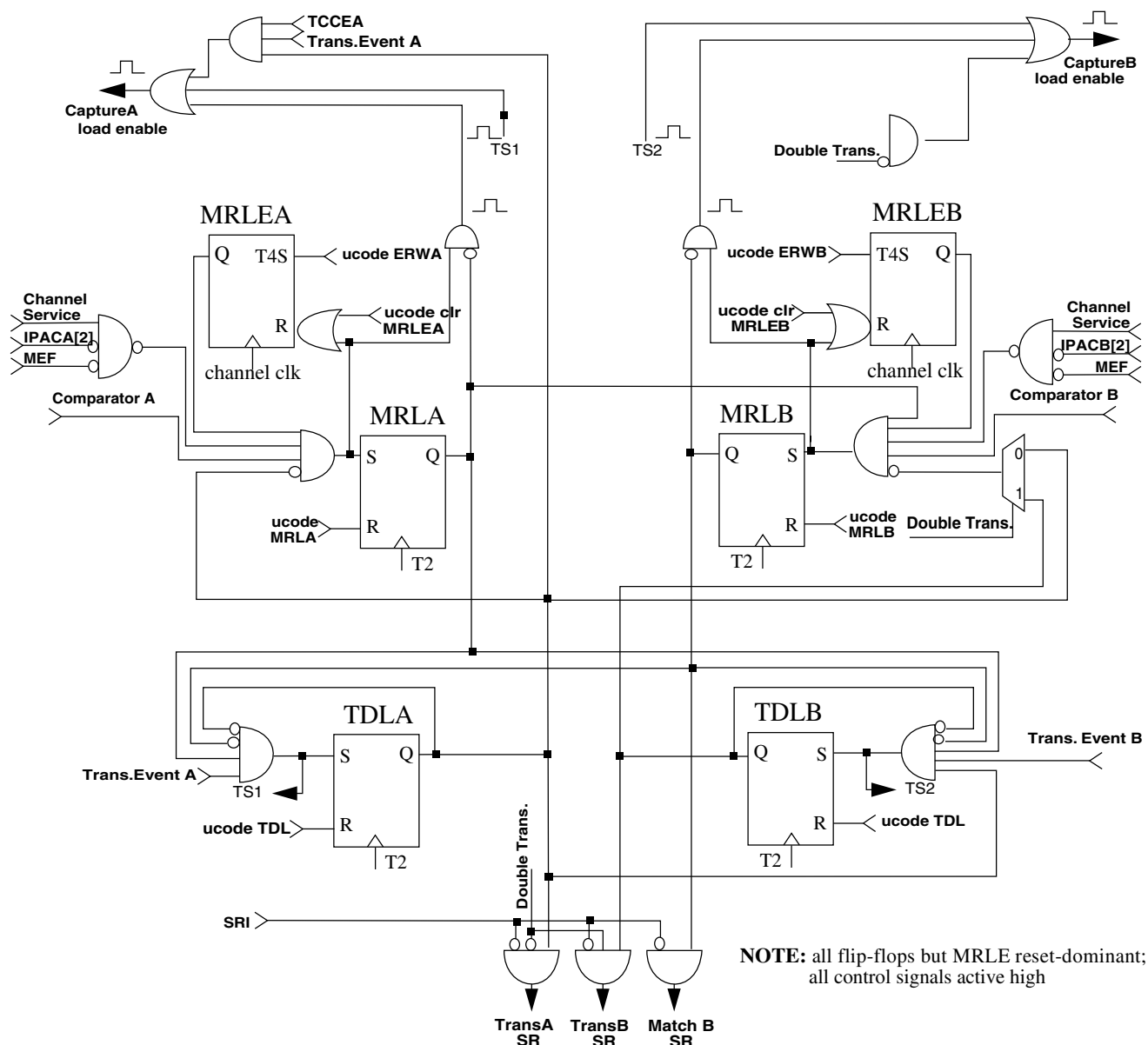


Figure 37-312. Ordered Modes with Match B Request (m2_o_st, m2_o_dt)

37.15.5.4.2.6 Single Match Modes (sm_st, sm_dt)

Single match modes support single or double transition with single match recognition. MRLB is never set, and MRLEB has no effect. The associated logic is shown in the following figure.

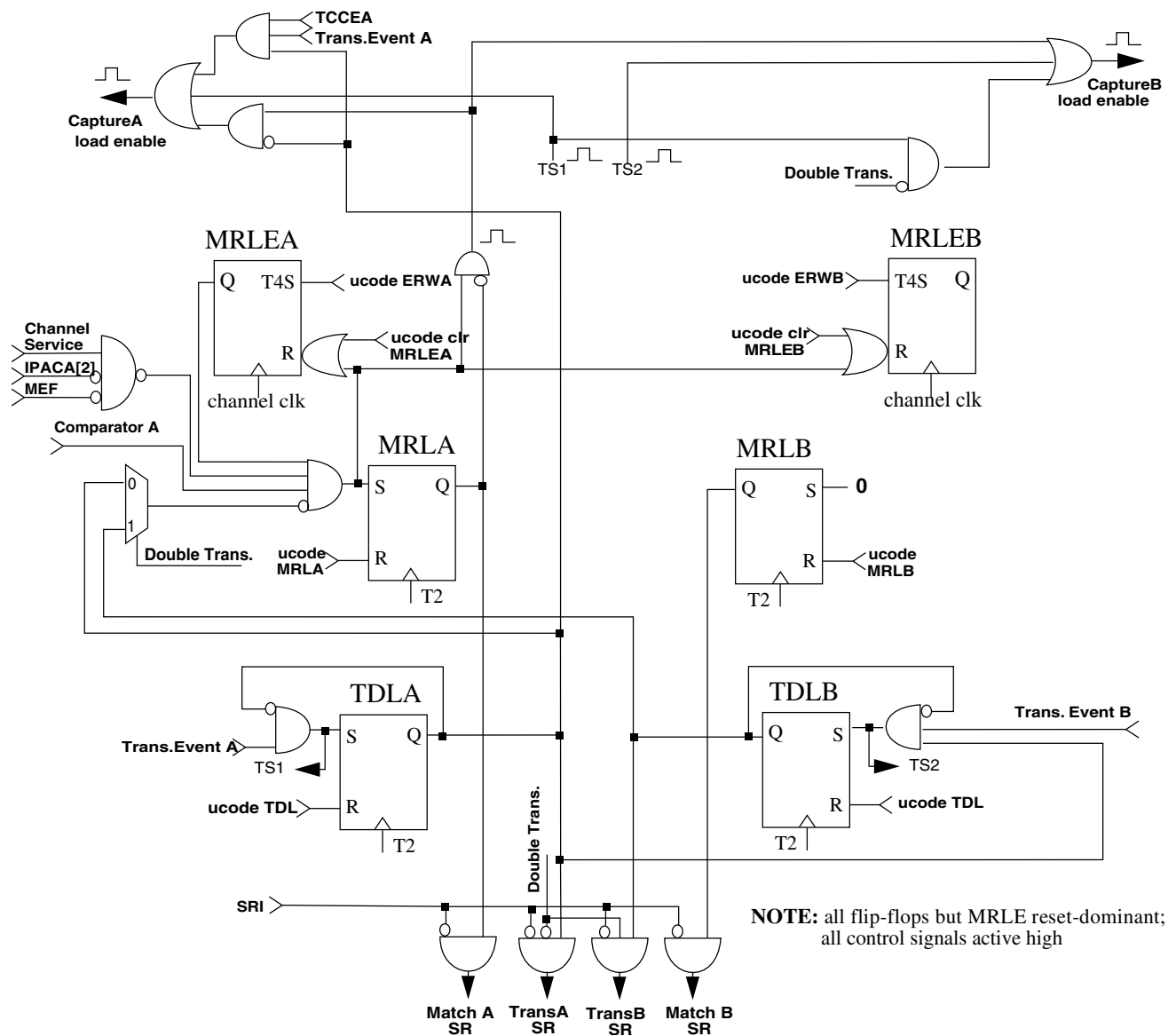


Figure 37-313. Single Match Modes (sm_st, sm_dt)

37.15.5.4.2.7 Single Match Enhanced Mode (sm_st_e)

This is an enhanced single transition and single capture mode, which provides non-filtered input captures in addition to the single capture, allowing one to measure the delay of the digital filter. In an output channel, it has the same functionality of sm_st (captures both time bases at once due to a match recognition). The associated logic is shown in the following figure.

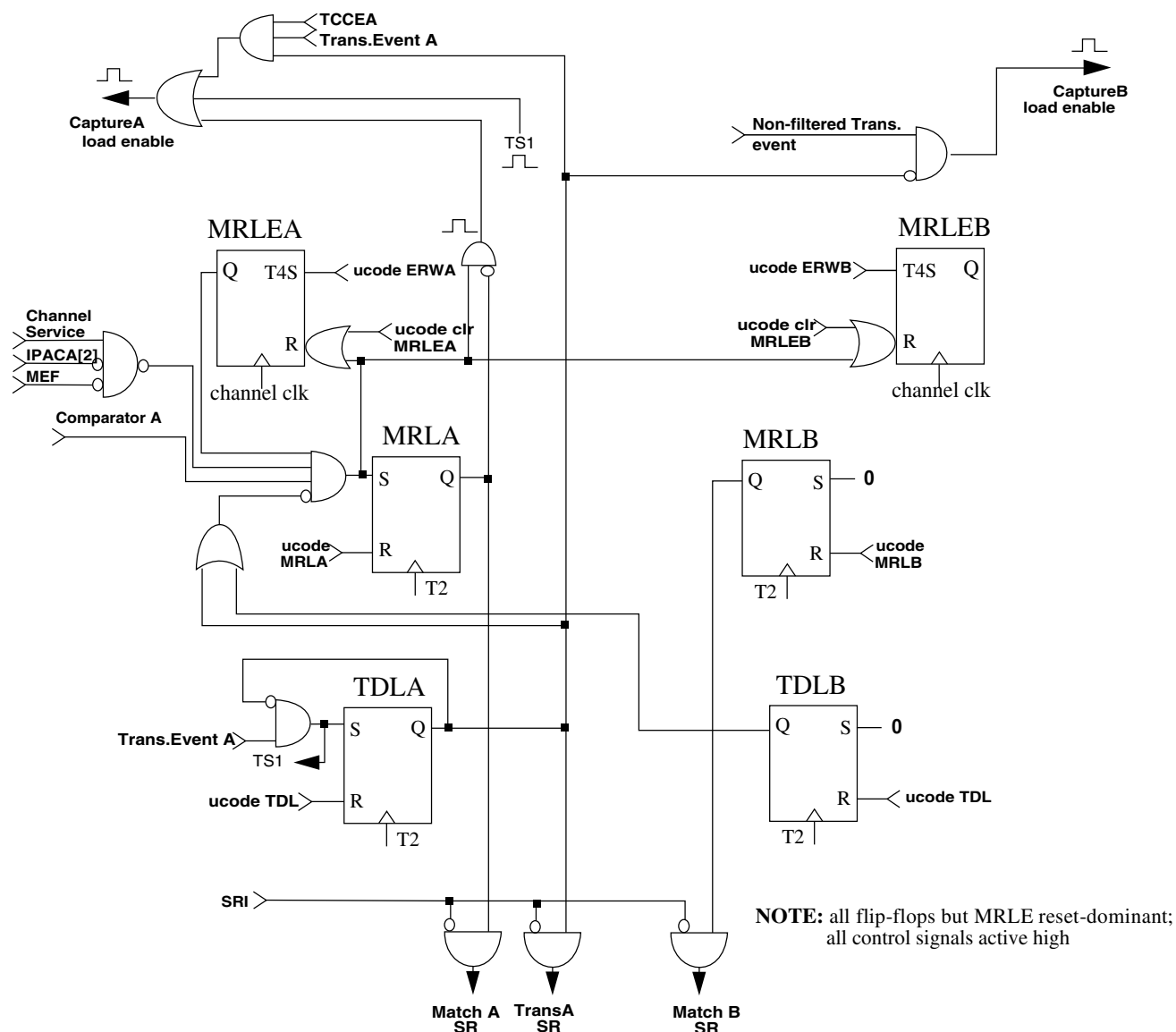


Figure 37-314. Single Match Enhanced Mode (sm_st_e)

37.15.5.4.3 Predefined Channel Modes on Input Signal Processing

When processing an input signal, the predefined channel modes can be classified in the following primary mode groups:

- Single Transition, Single Match: em_b_st, sm_st, sm_st_e
- Single Transition, Double Match: em_nb_st, bm_st, m2_st, m2_o_st
- Double Transition, Single Match: em_b_dt, sm_dt
- Double Transition, Double Match: em_nb_dt, bm_dt, m2_dt, m2_o_dt

In single transition modes, TDLA assertion may capture both time bases at once, while in double transition modes each transition captures its related time base in its related capture register.

Double transition is always ordered, i.e TDLB is enabled by TDLA and generates the service request.

The channel logic supports various input modes with combinations of single/double transition and single/double match, explained in the following subsections.

37.15.5.4.3.1 Either Match, Blocking, Single Transition (em_b_st)

On an input signal, this mode provides double timeout mechanism on a programmed transition edge with two timebases. The signal transition blocks both pending matches, indicating that no timeout condition occurred. The two match recognitions block each other, giving good separation in the entry table as to which match recognition caused the first timeout condition, and generating only one service request. Either match performs timebase captures which do not overwrite captures by transitions.

37.15.5.4.3.2 Either Match, Blocking, Double Transition (em_b_dt)

In double transition mode each transition is related to one match recognition. TDLA assertion captures its related timebase, blocks Match A and enables TDLB. TDLB assertion blocks Match B, captures its related timebase and generates a service request. Match recognitions block each other, so if there is a match timeout condition on TDLA, only one match service request is generated. This mode is good for qualifying two signal transitions by match timeout mechanisms, with one service request. Note that although TDLA assertion does not block Match B recognition, the value captured in CaptureA by TDLA assertion is not overwritten by this recognition. The second transition blocks Match B. Either match performs timebase captures which do not overwrite captures by transitions.

37.15.5.4.3.3 Either Match, Non Blocking, Single Transition (em_nb_st)

On an input signal, this is a double timeout mechanism of independent match recognitions of two different timebases. The match recognitions do not block each other, such that the microcode can check if one or two match recognitions occurred before their related signal transition. The signal transition detection (by IPACA) asserts TDLA, blocks both matches, captures both time bases and generates a transition service request, indicating that none of the two timeout conditions occurred. Any combination can be easily resolved by microcode (for example, signal transition after Match A and before Match B, or signal transition after both Match A and Match B).

Another possible use of this mode is allocating one match recognition for transition timeout and the other for another non-critical timed task, adding functionality to a single channel. Since the transition detection blocks both match recognitions, the match recognition of the other timed task is based on the fact that the comparator checks greater-equal conditions. It may be delayed if it occurs in the period between the signal transition detection (which blocks it) and the time TDLA is negated by microcode. If matches are enabled during the service, the same code can check if the match recognition of the timed task occurred in this period, by negating TDLA and writing to the CHAN register its own value (in order to update the MRLA flag in the branch logic).

37.15.5.4.3.4 Either Match, Non Blocking, Double Transition (em_nb_dt)

In this mode each transition is related to one match recognition, and the match recognitions are independent of each other. This mode can be used to give independent timeout conditions for the first and the second signal transition recognitions, and call service in any case of any timeout condition.

The first transition detection programmed in IPACA sets TDLA, captures its related timebase, blocks Match A recognition and enables TDLB assertion. The second transition detection programmed in IPACB sets TDLB, blocks Match B recognition, captures its related timebase and generates a service request. Any match recognition that occurs captures its related time base and generates a match service request, independent on the other match recognition.

37.15.5.4.3.5 Match B Request, Single Transition (m2_st)

On an input signal, this mode provides an open window filter for a single signal transition. MRLA assertion opens the window, and enables transition detection on TDLA from this time on. MRLB assertion blocks Match A (by negating MRLEA), providing conditional window opening, because transitions are indirectly blocked. It also generates service request, but if it happens after Match A it does not block transitions, providing a non-blocking timeout mechanism for the estimated signal transition time (typically it indicates a missing transition, or mis-prediction of the transition time).

Transitions can be detected from the microcycle following MRLA assertion. The Transition A detection asserts TDLA, blocks both matches, captures both timebases and generates service request.

Using this mode, the channel can replace software open window filtering of qualified transitions with the channel hardware window. The window opening and timeout can be scheduled for any of the two time bases or combination of them. Typically, Match A will be used to open a prediction window, and Match B will be used as a timeout condition

which does not close the prediction window. This configuration improves noise immunity from early signal transitions, and reduces the probability for blocking late signal transitions due to timeout mis-prediction.

Using these conditions, the microcode can easily resolve the state:

- If TDLA and MRLA are asserted and MRLB negated, signal transition is in the expected range.
- If MRLA and MRLB are both asserted, and TDLA is asserted, the signal transition had a timeout condition due to Match B mis-prediction.
- If MRLB is asserted and TDLA negated, a timeout condition occurred, and the expected signal transition had not occurred yet.
- If MRLA is negated and MRLB is asserted, the conditional window did not open at all (for example: a time window is open only after a specific angle, otherwise it is not opened).

37.15.5.4.3.6 Match B Request, Double Transition (m2_dt)

This mode is used as an open window filter for two signal transitions. In this case the Match A recognition opens the window (unless Match B recognition occurred first), and Match B recognition blocks Match A and generates a match service request. It is similar to m2_st, but in this case, it is the second transition that blocks Match B. MRLB assertion is a global timeout condition for the two pulses. Like m2_st, MRLB can conditionally eliminate the window from opening.

Using the TDLA, TDLB, MRLA and MRLB conditions, the microcode can easily resolve the state, in a similar manner as m2_st, with additional information on the second transition (TDLB).

37.15.5.4.3.7 Both Match Request, Single Transition (bm_st)

On an input signal, this is a double timeout mechanism on two different time bases. Both match recognitions must occur before the signal transition to generate a match timeout service request. Assertion of TDLA blocks both Match A and Match B recognitions, and captures both time bases, indicating there was no double timeout condition from both time bases.

Using the same timebase implements two timeout conditions, the first only sets its related MRL and the second generates a service request. Using these flags allows the microcode to check if one or both match recognitions precede the signal transition.

37.15.5.4.3.8 Both Match Request, Double Transition (bm_dt)

In this mode the first transition detection does not block matches, since both match recognitions are required to generate a match service request. The second transition detection asserts TDLB, blocks Match A and Match B, captures its related timebase and generates transition service request. In this mode, a Match A recognition which occurs after the assertion of TDLA does not capture a new value in CaptureA, to preserve the actual signal transition time. Assertion of TDLA, however, always captures its related timebase.

This mode allows putting a double match timeout condition on the second transition. Typically, a pulse trailing edge timing can be checked against two time bases, to indicate if the pulse has not ended when both MRLA and MRLB are asserted. When a transition service request is generated by TDLB assertion, the state of MRLA and MRLB indicates which timeout condition occurred, if any.

37.15.5.4.3.9 Ordered Mode with Match B Request, Single Transition (m2_o_st)

On an input channel, this mode provides a closing window filter for a single signal transition. Match A assertion captures its programmed time base in CaptureA, opens the filter window (enables assertion of TDLA), and enables assertion of MRLB. Match B recognition captures its related timebase, closes the window (disables assertion of TDLA) and generates a service request. Due to Match A and Match B ordering, the window is opened for at least one microcycle. Match B recognition indicates a window timeout condition which blocks late signal transitions, outside the prediction window. Transition detection blocks both matches, indicating the transition occurred inside the estimated window. Transitions can be detected from the microcycle following MRLA assertion until the microcycle on which MRLB is asserted. When TDLA is asserted inside the window range it disables both matches, captures both time bases and generates a transition service request.

Using this mode, the channel can replace software window filtering of qualified transitions with the channel hardware window. The window opening and closing time can be scheduled for any of the two time bases or a combination of them.

37.15.5.4.3.10 Ordered Mode with Match B Request, Double Transition (m2_o_dt)

In this mode the channel logic implements a window filter for two detected signal transitions. MRLA assertion captures its related timebase and enables the assertion of both TDLA and TDLB. MRLB assertion captures its related timebase and disables the assertion of both TDLA and TDLB. Transitions can be detected from the microcycle following MRLA assertion until the microcycle on which MRLB is asserted. The first

signal transition (following MRLA assertion) asserts TDLA, captures its related timebase and enables assertion of TDLB. The second signal transition detection asserts TDLB, blocks Match B, captures its related timebase and generates the service request.

If both signal transitions occur inside the scheduled window, Match B recognition is blocked. If one or both signal transitions do not occur inside the scheduled window, Match B recognition generates a match service request and blocks further transition detections. The microcode can resolve the state using MRLA, MRLB, TDLA and TDLB, which affect the entry point selection.

37.15.5.4.3.11 Single Match Enhanced Mode (sm_st_e)

This is an enhanced single transition and single match channel mode which provides timing information of the digital filter delay.

The CaptureA register captures the timebase selected by TBSA due to transition detection specified by IPACA or match recognition, as in sm_st mode. Initially, the CaptureB register continuously captures the unfiltered IPACB-selected signal transitions from the digital filter input, directly from the signal synchronizer. When an IPACA-qualified, filtered transition detection occurs, TDLA is set, MRLA assertion is blocked, and, in addition, captures into CaptureB are also blocked. On service, CaptureA and CaptureB (copied into ERTA and ERTB) holds the time of the qualified transition detection (ERTA), and the time of the last signal transition at the input of the digital filter (ERTB). Subtracting the time in ERTB from the time in ERTA provides the delay of the digital filter.

In a quiet environment, the two captures provide the accurate delay of the digital filter in granularity of two eTPU clocks. In a noisy environment, false transitions may be detected at the input of the digital filter due to the noise, and the delay measurement may be reduced, especially if IPACB selects both edge detection. The microcode can do sanity checks on this value to recognize noise effects (for example: calculated delay is less than the minimum delay of the digital filter).

Note

In Channel 0, if eTPU_TBR_TBCR_ENG n [AM]=01 (Angle Mode), the unfiltered input comes from TCRCLK input and the filtered input comes from the TCRCLK filter output. The edge is selected by IPACA/B, and is independent of the edge selection by eTPU_TBR_TBCR_ENG n [TCR2CTL].

37.15.5.4.3.12 Single Match, Single Transition (sm_st)

In this mode the channel logic is functionally back-compatible to a TPU3 single action channel, but a match or transition detection captures at once both timebases. The mode recognizes a single transition with single match timeout. Either TDLA or MRLA generates service request and captures both timebases. Assertion of TDLA blocks future assertions of MRLA.

37.15.5.4.3.13 Single Match, Double Transition (sm_dt)

In this mode, the first transition detection asserts TDLA, captures a timebase in CaptureA and enables TDLB. The second signal transition asserts TDLB, blocks Match A, captures a timebase in CaptureB and generates a service request.

Match A (before TDLB) captures into CaptureB the timebase selected by TBSA, in order not to overwrite the captured value of TDLA.

This mode is used for scheduling one timeout condition on two input signal transitions (pulse timeout).

37.15.5.4.4 Channel Modes on Output Signal Generation

Since channel logic can generate output signal transitions based on Matches, the channel can be viewed as working in the following primary mode groups for signal generation:

- Single Match: em_b_st, sm_st, sm_st_e, em_b_dt, sm_dt
- Double Match: em_nb_st, bm_st, m2_st, m2_o_st, em_nb_dt, bm_dt, m2_dt, m2_o_dt

The channel logic supports various match channel modes with single/double match, as explained in the following subsections.

37.15.5.4.4.1 Either Match, Blocking Modes (em_b_st, em_b_dt)

On an output signal these modes are useful when using two different time bases to set a required signal transition. The first match condition which is met sets a required pin action, captures both time bases, blocks any effects of the other recognition, and generates a service request. Because the first match recognition blocks the other, the microcode can get good separation in the function entry table as to which match caused the timeout first, and both time bases are captured, enabling the microcode to compare one timebase to the other at the moment of the match recognition. These modes can be used for:

- Scheduling a required pin action to the first match recognition of two different time bases.
- Canceling a programmed pin action scheduled on one time base by match on another timebase (as a consequence of [Table 37-353](#)). Microcode has to set the OPAC register of the canceling match to no-action and the OPAC register of the other match to the required pin action which may be blocked. If Match A is the canceling match, it blocks the pin action also in case of two matches at the same time, since it has priority in this case. If Match B is the canceling match, it does not block the pin action in case of two matches at the same time.

37.15.5.4.4.2 Either Match, Non Blocking Modes (em_nb_st, em_nb_dt)

On an output signal these modes are useful in combination with the ME bit set on the entry point, to define an interlaced operation. For example, each match recognition can set a pin action, and the second pin action is not sensitive to microcode latency (ME bit asserted). Example for usage is PWM interlaced function on which the latency is determined by the period and not the duty cycle.

Another possibility is using one match for pin actions and the other match for an unrelated timed task without pin action (double the functionality of a single channel).

37.15.5.4.4.3 Match B Request Modes (m2_st, m2_dt)

On an output signal, these modes can generate narrow pulses or do conditional pin actions. A conditional pin action means that the pin state is changed only if the match recognitions occurred in the correct order, since the Match B recognition which generates the service request also has priority over the pin action and blocks future Match A recognitions.

Setting OPACA to a desired pin action and OPACB to no-action, and using different time bases for Match A and Match B defines a conditional OPACA pin action which can be blocked by Match B recognition. For example, setting Match A on time and Match B on angle can limit the pin action to a maximum angle value.

When pulses are generated, the service is requested at the trailing edge of the pulse, after MRLB is asserted.

37.15.5.4.4.4 Both Match Request Modes (bm_st, bm_dt)

On an output signal, each match recognition can affect the pin state, and capture its programmed time base. This way the pin action can be programmed separately for both match recognitions. For example, both match recognitions can negate the signal, and

service request is generated after both conditions are met. This mechanism can set two conditions to do a required pin action, and the first recognition changes the signal, but service is called only after both conditions occur.

When using the same time base, these modes can generate narrow pulses in any required order. For example, in a PWM function, when duty cycle is below 50% the function can get service on the low time and program the pulse to the required duty cycle of the high time. When duty cycle is equal or above 50%, the function can get service on the high time and program a negative pulse with the width of the required low time. To switch between the two states the function can program once the same transition time to MatchA and Match B with a required pin action, and on the next service program double match for the new state.

Another usage is generating a required pin action on one programmed time and service request later on another time, after the second match recognition occurs, or capturing some timebase on one time and generating a required signal transition and service request later.

37.15.5.4.4.5 Ordered Modes with Match B Request (m2_o_st, m2_o_dt)

The order of the match recognitions imply that OPACA register programmed pin action always precede the OPACB register pin action. Setting OPACA to no-action, based on the greater-equal comparator, enables using Match A on one time base to delay the signal effect of Match B on the other time base. This method implements a conditional pulse extension or conditional delay on signal transition.

These modes can also be used for deferred pulse generation with microcode service request after its trailing edge (if Match A condition comes after Match B condition). Another option is having Match A recognition associated with output pin actions and Match B recognition for a timed microcode task which has to be scheduled at a programmed time which may be delayed by the Match A pin action.

37.15.5.4.4.6 Single Match Modes (sm_st, sm_dt, sm_st_e)

There is no difference between plain and enhanced Single Match modes on an output signal.

In this mode the channel logic is functionally back-compatible to a TPU3 single match output channel. Match A recognition generates service request and sets the pin state according to OPACA register. It captures at once the timebase selected by TBSA in CaptureA and the timebase selected by TBSB in CaptureB.

37.15.5.4.4.7 Match/Transition Pin Action Conflict Resolution

In output signals, matches and/or transitions automatically cause pin actions defined by the OPACA/B and/or IPACA/B channel control registers (see [Pin Control Registers](#)). Simultaneous matches/transitions may be associated with different, possibly contradictory, pin actions. These conflicts are resolved according to the table below.

If an OPACA/B=000 (no action) prevails over non-zero OPAC according to the table below, then if Match A/Transition A and Match B/Transition B occur simultaneously, no output pin action occurs, that is: a match on the action logic with OPAC=000 inhibits simultaneous actions of the other OPAC, if prevailing according to the following table. That also applies when output actions are caused by inputs (OPAC=1xx).

Table 37-353. Simultaneous Match Pin Action Priority

Channel Mode	Priority
em_nb_st / em_nb_dt	OPACA
em_b_st / em_b_dt	OPACA
bm_st / bm_dt	OPACA
m2_st / m2_dt	OPACB
m2_o_st / m2_o_dt	in these modes there is no possibility of simultaneous matches
user-defined	OPACB if M2BM1 = 1 and M1BM2 = 0, OPACA otherwise

37.15.5.4.5 Combining Input and Output Signals

The processing of input signal can be combined with output signal generation. A detected input transition can trigger an output signal edge, even without microcode intervention, by using OPAC options 1xx.

The channel set-up examples below show these two capabilities combined (see the figure below).

The **first example** implements a fast (no microcode intervention) short-circuit protection feedback mechanism for driving high-current output devices. The signal after the high-current driver feeds back to the channel input. The input signal is normally delayed from the output signal by the device turn-on delay. After the channel output turns on, the channel logic must check if the driver output (connected to the channel input) follows the driven value after the maximum device turn-on delay. If it does not, the driver output is probably shorted, and the channel output must be turned off immediately to avoid damaging the device.

Match A causes the output to be driven high (for simplicity the output and input signals are shown as positive logic). It also causes a transition A, because IPACA=100 and the input is still low. Match B occurs after the expected driver delay, and causes a service request. If the output is shorted, a Transition B occurs on Match B because IPACB=100. This will cause the output to go low immediately, because OPACB=100.

In the **second example**, an output pulse is generated from an input transition without microcode intervention. Match A opens a window for transitions and also enables Match B. A rising edge on input sets output high. On Match B the window closes, and input signal is checked: if sampled high, the output resets; otherwise it stays high.

In the **third example**, a pulse is generated depending on the value sampled on the input signal at a predetermined time. Match A samples the input signal, causing a Transition. Low level on input sets output low, otherwise it stays high. Match B sets output high. In both cases a service request is issued (microcode intervention), at the beginning and at the end of the pulse (Match B), if required (SRI := 0).

Note

When IPAC = 1xx, a match event can cause simultaneously a Match recognition and a Transition detection. Depending on the Channel Mode, these Match and Transition may have conflicting effects on other Transition/Match blocking or enabling. In these cases, blocking always prevails over enabling, effective on the next microcycle.

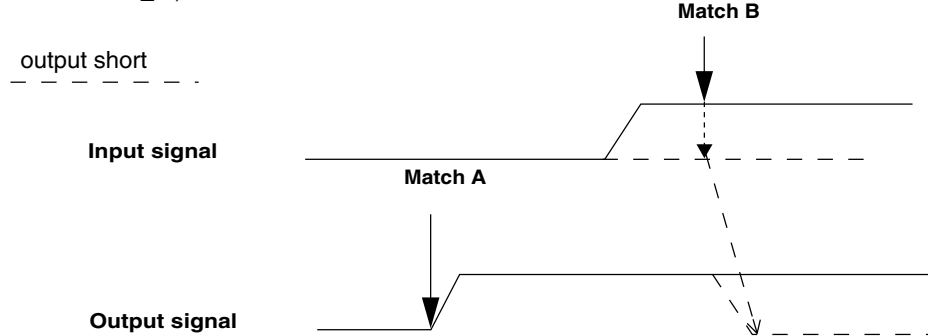
Functional description

Example 1: Short-circuit protection feedback

IPACA := 100; OPACA := 001; MatchA := output activate time;

IPACB := 100; OPACB := 100; MatchB := MatchA + max. high-current driver turn-on delay;

PDCM := bm_dt;

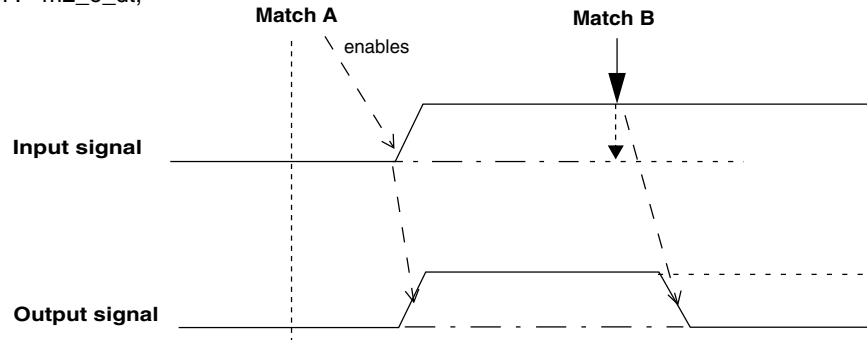


Example 2: Pulse generation on windowed input transition

IPACA := 001; OPACA := 101; MatchA := window open time;

IPACB := 101; OPACB := 100; MatchB := window close time, input sampling;

PDCM := m2_o_dt;



Example 3: Pulse generation on input sampling

IPACA := 100; OPACA := 100; MatchA := window open time, input sampling;

IPACB := 000; OPACB := 001; MatchB := window close time = MatchA + pulse width;

PDCM := em_nb_dt;

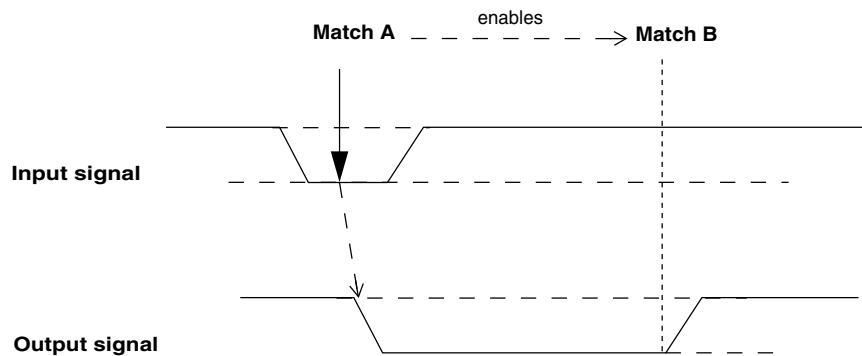


Figure 37-315. Input/Output combination

37.15.5.5 Channel Link

A channel can issue service requests to other channels through microcode, by assigning to the write-only microengine register LINK (refer to [LINK Register](#)) a value which specifies the target channel of the Link Service Request, as shown in the figure below.

Writing to the LINK register issues a link request to the target channel, setting its LSR flag. Each channel has its own LSR flag, which can be tested as a microcode branch condition (see [Conditional/unconditional branch](#)) and reset through the microcode field LSR (see [Clear Link Service Request](#)). The link branch condition samples, at the TST start, the value used to calculate the Entry Point.

Writing LINK with another channel target value in the same thread issues a Link Service Request to the new target, without negating the service request to the former one. This allows a channel to issue service requests to any number of channels, including itself. Neither LINK nor LSR microengine accesses are qualified by the CHAN register, i.e., they always access the serviced channel LINK and LSR, regardless of the value written in CHAN.

Channels can also receive link service requests from the Shared Subsystem Units (see [Shared Subsystems support](#)). These links are set by the interrupt signals from the shared subsystem and are level-sensitive, that is, they keep on requesting link service as long as the interrupt is active. Therefore, it is not enough to clear the link service request through the LSR instruction field: one must clear beforehand the respective shared subsystem unit interrupt source.

If microcode executes an instruction with field LSR=0 (clear Link Service Request), the link branch condition is cleared. However, the link service request itself is cleared only if no link from other channel was received by the serviced channel during the same thread³⁰. Note that this applies only to links that come from the other engine or the same channel: if only Shared Subsystem link was received during the thread, the link service is also cleared with field LSR=0. If microengine clears LSR of its channel and, simultaneously, Link Service Request is issued to the current serviced channel, the branch condition is cleared but the link service request remains pending.

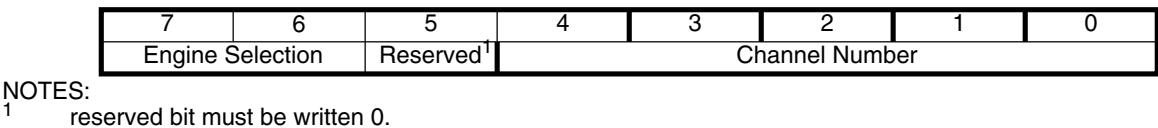


Figure 37-316. Microengine LINK Register

30. That can only happen if the link service request came from the other Engine, or from the serviced channel itself.

A channel can issue Link Service Requests to channels in any of two Engines, determined by the LINK register field Engine Selection (2 bits), as shown in the following table. In a Single Engine eTPU, Link is ignored when sent to the other engine, or engine 2.

Table 37-354. LINK Engine Selection

Engine Selection	Description
00	this Engine
01	Engine 1
10 ¹	Engine 2
11 ¹	the other Engine

1. ignored in Single Engine eTPU.

There is no explicit hardware mechanism to allow the thread to identify the link service request source, but it can be done by a software protocol using SDM parameters. All links are negated on reset.

37.15.5.6 Enhanced Digital Filter - EDF

The EDF eliminates passing of signal transitions which are caused by noise. Its purpose is to eliminate false transition service requests caused by noise pulses which are shorter than a programmed width.

See the chip-specific eTPU information for connections between eTPU and input glitch filters.

The EDF has three modes of operations, selected by the CDFC field in [eTPU Engine 1 Configuration Register \(eTPU_SCR_ECR_ENG1\)](#) and [eTPU Engine 2 Configuration Register \(eTPU_SCR_ECR_ENG2\)](#). These modes offer selections of trade-off between noise immunity and signal latency. CDFC also allows the filter to be bypassed. [Table 37-355](#) gives an example of minimum detected signal pulse and maximum filtered noise pulse in the three EDF operation modes. In Angle Mode, if AM=01, the EDF in channel 0 is replaced with the digital filter and synchronizer of the TCRCLK signal. In this mode, channel 0 works in combination with the Angle Counter logic, and their operation is fully synchronized.

Following subsections provide the functional description of the eTPU channel digital filter.

37.15.5.6.1 Two-Sample mode

In this mode the EDF works like the TPU2/3 digital filter. It uses the filter clock which is the eTPU clock divided by (2, 4, 8,..., 256) as a sampling clock. The filter clock is selected by the FPSCK field in [eTPU Engine 1 Configuration Register \(eTPU_SCR_ECR_ENG1\)](#) and [eTPU Engine 2 Configuration Register \(eTPU_SCR_ECR_ENG2\)](#). The EDF compares two consecutive samples. If both samples have the same value, the input signal state is updated. Note that when the FPSCK field selects the eTPU clock divided by two, the EDF works like the TPU1 four-clock digital filter.

37.15.5.6.2 Three-Sample mode

In this mode, like in the TPU2/3 mode, the EDF uses the filter clock as a sampling clock. The EDF compares three consecutive samples. If all three samples have the same value, the input signal state is updated.

The Three-Sample mode gives more signal latency than the Two-Sample mode, but also better noise immunity and better ratio between minimum detected signal pulse to maximum filtered noise pulse. When a certain filter clock frequency is selected for Two-sample mode, double filter clock frequency can be selected to get better latency in Three-sample mode.

37.15.5.6.3 Continuous mode

In this mode the EDF compares all the values sampled at the rate of eTPU clock divided by two, between two consecutive filter clock pulses. If the signal is continuously stable for the entire digital filter clock period (i.e all the samples have the same signal value), the input signal state is updated.

This method gives the same latency and the same ratio between minimum detected signal pulse to maximum filtered noise pulse, as the Two-Sample mode, as long as there is no noise. Each sampled noise delays the signal transition detection by at least a whole digital filter clock period.

The Continuous mode gives the best noise immunity by comparing multiple samples of the noise. On the other hand, when a short noise pulse appears in the middle of the filter clock period at the same time of a real signal transition, the Continuous mode may reject a real signal transition and delay the response to the first filter clock period in which the signal is continuously stable. This may add to the latency and also to the minimum detected signal pulse in a noisy environment.

37.15.5.6.4 Bypass Mode

In bypass mode the signal that feeds the edge detection comes directly from the output of the synchronizer, not filtered. Bypass mode automatically makes the channel logic work in T2/T4 timing mode (see [Channel timing modes](#)).

37.15.5.6.5 Filter clock prescaler

Each channel configured as an input has an associated synchronizer followed by a digital filter connected to the signal that samples signal transitions. After reset, the digital filter filters out high and low pulse widths smaller than the period of two eTPU clocks with eTPU_SCR_ECR_ENG n field FCSS=0, or 1 eTPU clock with FCSS=1 (see [eTPU Engine 1 Configuration Register \(eTPU_SCR_ECR_ENG1\)](#) and [eTPU Engine 2 Configuration Register \(eTPU_SCR_ECR_ENG2\)](#)), preventing these transitions from being input to the transition detect logic. For FPSCK=0 and FCSS=0, the synchronizer and digital filter are guaranteed to pass pulses that are as wide as or wider than four eTPU clocks, meaning a minimum period of 8 eTPU clocks. These figures are halved by setting FCSS=1. By changing the FPSCK field in [eTPU Engine 1 Configuration Register \(eTPU_SCR_ECR_ENG1\)](#) and [eTPU Engine 2 Configuration Register \(eTPU_SCR_ECR_ENG2\)](#) the user can select a lower clock rate for the filter signal to define wider valid pulses and filter out wider noise pulses. The filter prescaler clock control is a division of the eTPU clock. To guarantee pulse detection by the digital filter, the pulse must cover at least the stated number of samples at the filter clock rate. For example, a two sample digital filter must sample two points in the pulse to detect it. The table below shows the minimum guaranteed detected pulse width and the maximum filtered noise pulse width. The table refers only to the digital filter operation. The external pulses may have to be wider (to ensure detection) or narrower (to ensure filtering) depending on the rise/fall delay differences in the MCU receivers and internal logic. Delays introduced by synchronizer, filter and edge detection logic are explained in [Input/Output Signal Delays](#).

The TCRCLK signal also has a synchronizer and a filter. The FCSS bit has no influence on the clock prescaler of the TCRCLK filter, which works like the channel filter with FCSS=0.

Table 37-355. Pulse widths and delays

Filter control (FPSCK)		Filter clock period ¹	Min. width guaranteed detected / max. width filtered (min. filter delay / max. filter delay) ¹	
FCSS = 0	FCSS = 1		Two-Sample or Continuous mode	Three-Sample or Integrator ² mode
not avail.	000	1	2 / 1 (2 / 3)	3 / 2 (3 / 4)
000 ³	001	2	4 / 2 (3 / 3)	6 / 4 (5 / 5)

Table continues on the next page...

Table 37-355. Pulse widths and delays (continued)

Filter control (FPSC)		Filter clock period ¹	Min. width guaranteed detected / max. width filtered (min. filter delay / max. filter delay) ¹	
FCSS = 0	FCSS = 1		Two-Sample or Continuous mode	Three-Sample or Integrator ² mode
001	010	4	8 / 4 (5 / 7)	12 / 8 (9 / 11)
010	011	8	16 / 8 (9 / 15)	24 / 16 (17 / 23)
011	100	16	32 / 16 (17 / 31)	48 / 32 (33 / 47)
100	101	32	64 / 32 (33 / 63)	96 / 64 (65 / 95)
101	110	64	128 / 64 (65 / 127)	192 / 128 (129 / 191)
110	111	128	256 / 128 (129 / 255)	384 / 256 (257 / 383)
111	not avail.	256	512 / 256 (257 / 511)	768 / 512 (513 / 767)

1. This table shows pulse widths and delays in number of periods of the eTPU clock.
2. Integrator mode is available for TCRCLK filtering only, see [TCRCLK digital filter](#).
3. These delays also apply to the TCRCLK filter when TCRCF selects eTPU clock/2 as filter clock.

Note

The TCRCLK filter is clocked as if FCSS=0, always dividing eTPU clock /2 using FPSC, regardless of the FCSS field.

37.15.5.7 Channel timing modes

Channels can work on two different timing schemes, defining the period of channel clocking, tied to T2 and T4 microengine phases, as explained in subsections below. Microengine T2 and T4 phases are shown in [Microcycle and I/O timing](#).

37.15.5.7.1 T2 channel timing

In T2 timing mode the channel event state can only be updated every two eTPU clocks (see [Figure 37-346](#)). Pin state, TDLs, MRLs and Capture registers are updated on the microengine's T2 clock phase. MRLE clears also happen on T2, but MRLE setting occurs on T4, together with the Match register write by microcode (see [Write Channel Match and UDCM Registers](#)).

Channels work in T2 timing mode when either the T4DIS field in [eTPU Channel n Configuration Register for Engine 1 \(eTPU_CCCR_CnCR_ENG1\)](#) and [eTPU Channel n Configuration Register for Engine 2 \(eTPU_CCCR_CnCR_ENG2\)](#) is 1 or all the following conditions are true:

- The TCR1CS field in [eTPU Engine 1 Time Base Configuration Register \(eTPU_TBR_TBCR_ENG1\)](#) and [eTPU Engine 2 Time Base Configuration Register \(eTPU_TBR_TBCR_ENG2\)](#) is 0.
- The Enhanced Digital Filter is not configured as bypass (see [Enhanced Digital Filter - EDF](#)).
- The FCSS field in [eTPU Engine 1 Configuration Register \(eTPU_SCR_ECR_ENG1\)](#) and [eTPU Engine 2 Configuration Register \(eTPU_SCR_ECR_ENG2\)](#) is 0 (see [Filter clock prescaler](#)).

37.15.5.7.2 T2/T4 channel timing

In T2/T4 timing mode the channel event state can be updated on any eTPU clock (see [Figure 37-347](#)). Pin state, TDLs, MRLs, MRLEs, and Capture registers are updated either on microengine's T2 or T4 clock phases. MRLE clears can happen on T2 or T4, but MRLE setting occurs on T4 only, together with the Match register write by microcode (see [Write Channel Match and UDCM Registers](#)).

Channels work in T2/T4 timing mode when the T4DIS field in [eTPU Channel n Configuration Register for Engine 1 \(eTPU_CCCR_CnCR_ENG1\)](#) and [eTPU Channel n Configuration Register for Engine 2 \(eTPU_CCCR_CnCR_ENG2\)](#) is 0 and either one of the following conditions are true:

- The TCR1CS field in [eTPU Engine 1 Time Base Configuration Register \(eTPU_TBR_TBCR_ENG1\)](#) and [eTPU Engine 2 Time Base Configuration Register \(eTPU_TBR_TBCR_ENG2\)](#) is 1.
- The Enhanced Digital Filter is configured as bypass (see [Enhanced Digital Filter - EDF](#)).
- The FCSS field in [eTPU Engine 1 Configuration Register \(eTPU_SCR_ECR_ENG1\)](#) and [eTPU Engine 2 Configuration Register \(eTPU_SCR_ECR_ENG2\)](#) (see [Filter clock prescaler](#)).

T2/T4 timing mode demands special care on channel microcode that reprograms a match in a thread with matches enabled (entry point ME=1, see the “Entry Points” section). When the match register is written and the MRLA/B flag(s) is(are) cleared in the same instruction, a match can occur after MRLA/B is cleared, together with the set of respective MRLE. In this case, both MRL and MRLE will be set, and a match service request will occur if enabled. However, the match happened on the old match value, not on the new (reprogrammed) one. In order to prevent this ambiguity to the code that services the match, it is advisable to clear the MRLE (besides MRL) together with the

match reprogramming, avoiding the match on the old value to occur while the new match value is being written. The set of the MRLE due to match reprogramming prevails over the MRLE clear, thus allowing the new programmed match to occur in the future.

A similar case occurs when a match is blocked by a transition, the blocking TDL is cleared and the blocked match is reprogrammed at the same instruction: in T2/T4 timing mode: a match can occur on the old match value. The solution is the same as the case above, that is, clear the respective MRLE in the same instruction.

37.15.6 Time bases

Each eTPU engine has two Time Counter Registers, TCR1 and TCR2. They provide 24-bit time bases, shared by all 32 channels. Any channel can use both time bases to:

- Match channel's internal registers MatchA or MatchB;
- Capture time base value to channel's internal registers, CaptureA and/or CaptureB, when a match recognition or an Input transition detection occurs. For more information on channel events refer to [Enhanced Channels](#).

The TCR1 and TCR2 counters are accessible by the microcode for read and write operations. Its current value is used for getting the current time, and the captured values are used for channel relative time calculations of future events. When they are written at the same time they are incremented from any clock source, the written value prevails.

TCR1 with `eTPU_TBR_TBCR_ENGn[TCR1CS]=0` and TCR2 values are updated in T2 and read in T4 (see [Microcycle and I/O timing](#)). TCR1 can also work at full-speed eTPU clock, and so be updated on both T2 and T4, when `eTPU_TBR_TBCR_ENGn[TCR1CS]=1`. Both TCR1 and TCR2 values can be imported from or exported to the STAC bus. When their values are imported (STAC Clients), these registers are written from the STAC bus and can only be read by microcode.

The TCR2 counters between the two Engines are out of phase by 1 eTPU clock, even when Time Bases are shared between them through STAC. It also applies to TCR1 counters if `eTPU_TBR_TBCR_ENGn[TCR1CS]=0`, but they can be in phase otherwise.

37.15.6.1 Timer Count Register 1 - TCR1

TCR1 can be used in the following modes:

- Internally Clocked Mode

Functional description

- Externally Clocked Mode
- STAC Bus Client Mode

The host program can read TCR1 time base through [eTPU Engine 1 Time Base 1 \(TCR1\) Visibility Register \(eTPU_TBR_TB1R_ENG1\)](#) and [eTPU Engine 2 Time Base 1 \(TCR1\) Visibility Register \(eTPU_TBR_TB1R_ENG2\)](#).

The TCR1 bus runs through all the local engine channels. In channels which select TCR1 as MatchA and/or MatchB source, when its value is greater or equal to the programmed match value, a Match A and/or Match B event occurs on that channel. A recognized match event sets its related Match Recognition Latch 1 or 2, and according to the Predefined Channel Modes (PDCM) it may generate a channel service request. For details on eTPU channels see [Enhanced Channels](#).

37.15.6.1.1 Externally Clocked Mode

TCR1 can be driven externally by the TCRCLK input, after the digital filter. The TCR1 clock source is configured by the TCR1CTL bit, as shown in the following figure. For more information on clock source selection, see [eTPU Engine 1 Time Base Configuration Register \(eTPU_TBR_TBCR_ENG1\)](#) and [eTPU Engine 2 Time Base Configuration Register \(eTPU_TBR_TBCR_ENG2\)](#).

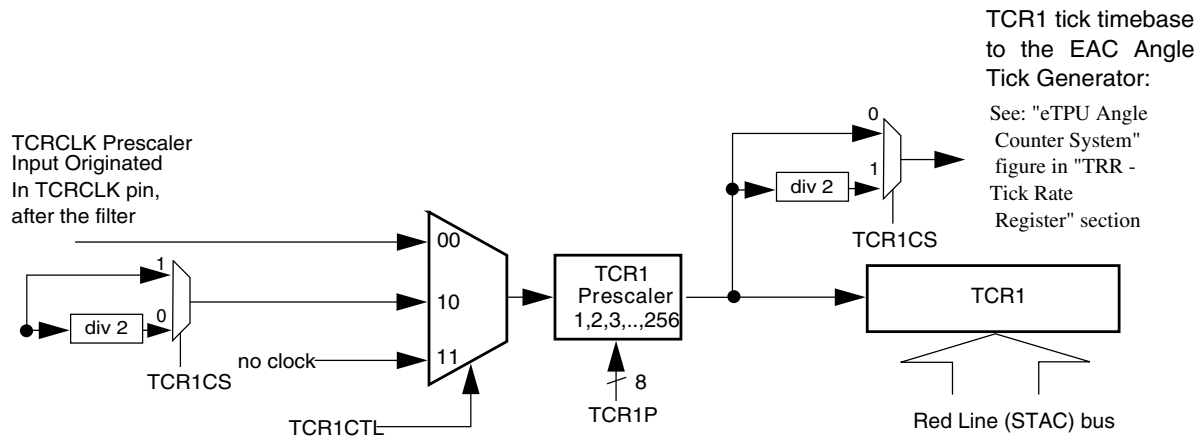


Figure 37-317. TCR1 clock selection

37.15.6.1.2 Internally Clocked Mode

TCR1 can be driven by the eTPU clock or eTPU clock divided by 2, before the prescaler. TCR1CTL can also be used to freeze TCR1 clock independently of TCR2 (unlike GTBE).

37.15.6.1.3 TCR1 clock prescaling

Any clock source selected by TCR1CTL is prescaled by a factor of 1 to 256, selected by the TCR1P field in [eTPU Engine 1 Time Base Configuration Register \(eTPU_TBR_TBCR_ENG1\)](#) and [eTPU Engine 2 Time Base Configuration Register \(eTPU_TBR_TBCR_ENG2\)](#). For more information on prescaler configuration see [eTPU Engine 1 Time Base Configuration Register \(eTPU_TBR_TBCR_ENG1\)](#) and [eTPU Engine 2 Time Base Configuration Register \(eTPU_TBR_TBCR_ENG2\)](#). The TCR1 Prescaler resets when the eTPU GTBE input is negated. After reset, it starts counting up to TCR1P when the eTPU GTBE input is asserted. When TCR1 increments (the eTPU GTBE input = 1), the prescaler starts a new count and the new TCR1P becomes effective. When TCR1 is written by microcode, the prescaler is reloaded with TCR1P and it becomes effective, if the eTPU GTBE input is asserted.

37.15.6.1.4 STAC Bus Client Mode

In this mode the TCR1 register is continuously updated from the STAC bus, and the clock selection and prescaling logic becomes ineffective. It is not writable by the microcode, and when read, it reflects the STAC bus imported value. The use of EAC is not allowed in client mode. This mode is configured through [eTPU Engine 1 STAC Configuration Register \(eTPU_TBR_STAC_ENG1\)](#) and [eTPU Engine 2 STAC Configuration Register \(eTPU_TBR_STAC_ENG2\)](#).

37.15.6.1.5 STAC Bus Server Mode

TCR1 bus can be exported to the STAC bus as a server, providing time information to other peripherals. This mode is configured through [eTPU Engine 1 STAC Configuration Register \(eTPU_TBR_STAC_ENG1\)](#) and [eTPU Engine 2 STAC Configuration Register \(eTPU_TBR_STAC_ENG2\)](#).

37.15.6.2 Timer Count Register 2 - TCR2

The TCR2 is a 24-bit counter which can be used in the following modes:

- Pin Transition Mode: Count the rise, fall or both transitions of TCRCLK signal.
- Angle Clock Mode: Count internal tooth angle in combination with the eTPU Angle Counter (EAC) hardware which implements an Angle PLL, and generates angle information to the channels. This mode is targeted for angle based applications.
- STAC (STAC) Bus Client Mode: TCR2 is driven by an external source (see [STAC Bus Client mode](#)).

- Gated Mode: Count with rate derived from the eTPU clock divided by eight. The TCRCLK signal is used to gate this count, enabling pulse accumulator operations.
- Internally Clocked Modes: TCR2 is driven by internal clock, with count rate of eTPU clock divided by eight.

All clock sources pass through a prescaler. In addition, the TCR2 count can be originated from the EAC which is a hardware angle clock and angle counter. Figure 37-318 shows the diagram for TCR2 clock control. When TCR2 is not driven by the EAC or STAC, the eTPU_TBR_TBCR_ENGn[TCR2CTL] field selects the clock source, also allowing TCR2 to be frozen independently of TCR1 (see eTPU Engine 1 Time Base Configuration Register (eTPU_TBR_TBCR_ENG1) and eTPU Engine 2 Time Base Configuration Register (eTPU_TBR_TBCR_ENG2)). When in Angle Mode, TCR2CTL selects the TCRCLK edge sensitivity.

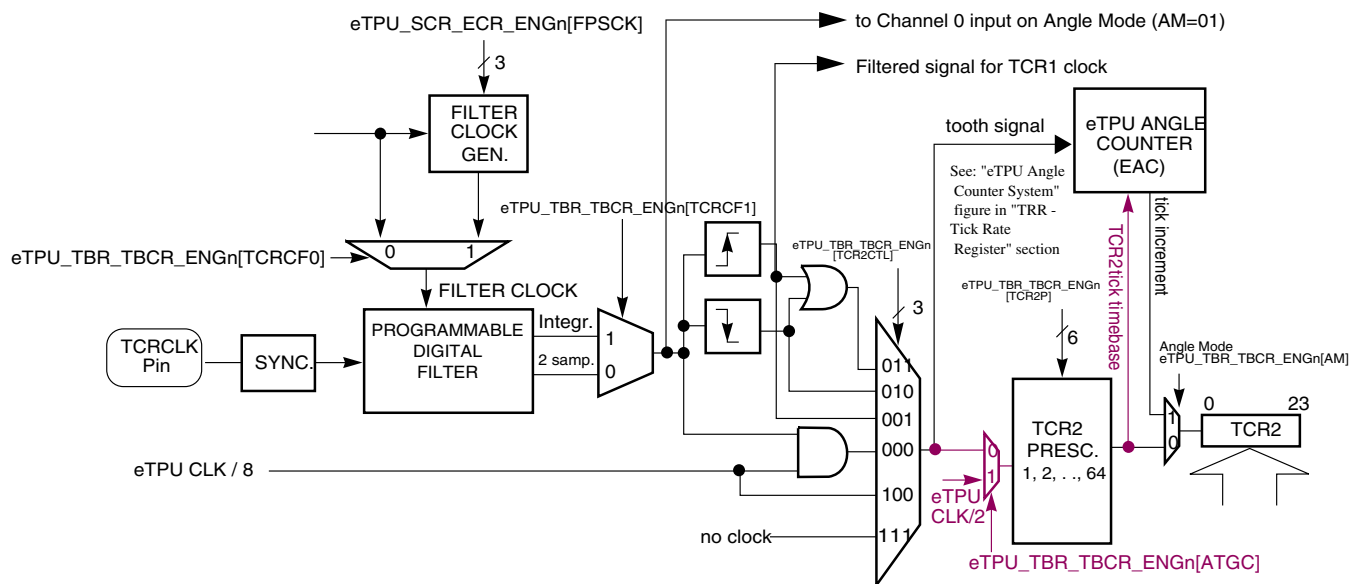


Figure 37-318. TCR2 clock control

The TCRCLK signal input is passed through a synchronizer and a programmable digital filter. In Angle Mode with AM=01, synchronizer and filter are also used in Channel 0, replacing its input synchronizer and filter, to get the same timing in the EAC and Channel 0. The TCRCLK synchronizer is an improved filter that provides best latency while maintaining proper noise filtering (see the field TCRCF in eTPU Engine 1 Time Base Configuration Register (eTPU_TBR_TBCR_ENG1))

The TCR2 bus runs through all the local engine channels. It transitions on clock T2 (see Microcycle and I/O timing). In channels which select TCR2 as MatchA and/or MatchB source, TCR2 value is compared against MatchA and/or MatchB registers. A recognized

match event sets its related Match Recognition Latch 1 or 2, and according to the Predefined Channel Modes (PDCM) it may generate a channel service request. For details on eTPU channels see [Enhanced Channels](#).

The TCR2 counter is accessible by the microcode for read and write operations. Its current value is used for getting the current counter value (representing signal transitions, time or angle), and the captured values are used for channel relative count calculations of future events. The TCR2 value is readable to the host through [eTPU Engine 1 Time Base 2 \(TCR2\) Visibility Register \(eTPU_TBR_TB2R_ENG1\)](#) and [eTPU Engine 2 Time Base 2 \(TCR2\) Visibility Register \(eTPU_TBR_TB2R_ENG2\)](#). When the TCR2 bus value is imported from the STAC bus (STAC client mode), TCR2 is not writable by the microcode, and read access from the microcode or from the host reflect the imported TCR2 value.

37.15.6.2.1 TCR2 clock prescaling

Except in Angle Mode, any clock source selected by TCR2CTL is prescaled by a factor of 1 to 64, selected by `eTPU_TBR_TBCR_ENGn[TCR2P]`. For more information on prescaler configuration, see [eTPU Engine 1 Time Base Configuration Register \(eTPU_TBR_TBCR_ENG1\)](#) and [eTPU Engine 2 Time Base Configuration Register \(eTPU_TBR_TBCR_ENG2\)](#). The TCR2 Prescaler resets when the eTPU GTBE input is negated. After reset, it starts counting up to TCR2P when the eTPU GTBE input is asserted. When TCR2 increments (the eTPU GTBE input = 1), the prescaler starts a new count and the new TCR2P becomes effective. When TCR2 is written by microcode, the prescaler is reloaded with TCR2P and it becomes effective, if the eTPU GTBE input is asserted.

The counter that divides the eTPU clock by 8 before the prescaler also resets when the eTPU GTBE input is negated, or when TCR2 is written by microcode.

In Angle Mode the TCR2 prescaler is not used to directly increment TCR2, but can be used, selected by programming `eTPU_TBR_TBCR_ENGn[ATGC]=1`, as a clock source for the EAC generation of the angle ticks which increments TCR2. In this case, the TCR2 prescaler input is eTPU clock/2, independently of TCR2CTL. See [Generating the angle ticks](#) for details.

37.15.6.2.2 TCR2 Gated mode

TCR2 Gated mode is selected in the `eTPU_TBR_TBCR_ENGn[TCR2CTL]` field. In this mode the TCRCLK signal enables or disables transfer of the eTPU clock divided by 8 to the TCR2 prescaler. By programming the prescaler, TCR2 can run at rates from eTPU clock divided by eight down to eTPU clock divided by 512, in steps of eight eTPU clock

divisions. For more information, see [eTPU Engine 1 Time Base Configuration Register \(eTPU_TBR_TBCR_ENG1\)](#) and [eTPU Engine 2 Time Base Configuration Register \(eTPU_TBR_TBCR_ENG2\)](#).

37.15.6.2.3 TCR2 Signal Transition modes

These modes are selected when eTPU_TBR_TBCR_ENG n [TCR2CTL] is set to rise, fall or "rise-and-fall". In these modes the TCRCLK signal is the TCR2 clock source, and its maximum transition rate depends on the TCRCLK digital filter mode of operation. The TCRCLK digital filter can be programmed to use the eTPU clock divided by two, or use the same filter clock of the channels, controlled by eTPU_TBR_TBCR_ENG n [TCRCF]. It contains an up-down counter which operates as a digital integrator, optimizing signal latency in the selected mode and clock rate.

When eTPU clock divided by two is selected, the synchronizer and the digital filter are guaranteed to pass pulses that are wider than four eTPU clocks (two filter clocks). Otherwise the TCRCLK is filtered with the same filter clock as the channel input signals. For details on TCRCLK and channels digital filter control, see [eTPU Engine 1 Time Base Configuration Register \(eTPU_TBR_TBCR_ENG1\)](#), [eTPU Engine 2 Time Base Configuration Register \(eTPU_TBR_TBCR_ENG2\)](#), and [Enhanced Digital Filter - EDF](#).

37.15.6.2.4 STAC Bus Client mode

In this mode the TCR2 register is continuously updated from the STAC bus, and the clock selection and prescaling logic becomes ineffective. It is not write accessible for the microcode, and when read, it reflects the STAC bus imported value. The use of EAC is not allowed in client mode. This mode is configured through [eTPU Engine 1 STAC Configuration Register \(eTPU_TBR_STAC_ENG1\)](#) and [eTPU Engine 2 STAC Configuration Register \(eTPU_TBR_STAC_ENG2\)](#).

37.15.6.2.5 STAC Bus Server mode

When TCR2 bus is exported to the STAC bus as a server, it can provide either time or angle bus to other peripherals, according to its operation mode. This mode is configured through [eTPU Engine 1 STAC Configuration Register \(eTPU_TBR_STAC_ENG1\)](#) and [eTPU Engine 2 STAC Configuration Register \(eTPU_TBR_STAC_ENG2\)](#). To provide sequential update of the STAC clients, the Angle tick rate must not be faster than the STAC programmed update rate. This requirement puts a limitation on the angle clock count rate on high rate mode. In this case the Angle and Angle Fraction accumulator (see [Angle Tick Generator](#) and [Figure 37-324](#)) are advanced at rate of eTPU clock divided by eight. Therefore, the STAC update rate for the Angle Bus must not be slower than eight eTPU clocks.

37.15.6.2.6 TCR2 bus in Angle Clock mode

In this mode the TCR2 counter operates as part of the eTPU Angle Counter (EAC). The TCR2 bus value reflects this angle representation in which it counts Angle Ticks. Angle Mode is selected when `eTPU_TBR_TBCR_ENGn[AM]=1`.

Note that when TCR2 works in Angle Mode, it does not count directly from the TCR2 clock input which indicates tooth signal transition. Its Angle counter is controlled by the Count Control and High Rate logic (see [Count Control and High Rate Logic](#)), which provides the interpolated pin position, and handle cases of missing tooth, acceleration, de-acceleration and mechanical corrections.

The EAC uses the TCRCLK signal to get the tooth transition indications. The `eTPU_TBR_TBCR_ENGn[TCR2CTL]` field has to be set for the appropriate tooth edge detection rise, fall, "rise-and-fall" or none. TCR2 count clock comes from the EAC control and not directly from the physical tooth. This way the EAC control processes the signal transitions and handles missing teeth and flywheel mechanical corrections. Note that when TCR2CTL selects "none" for tooth edge selection, the TCR2 is not necessarily frozen, but can still be incremented by the EAC logic.

In Angle Mode, eTPU channel 0, 1 or 2 operation is combined with the EAC operation. When channel 0 is selected for EAC operation, the TCRCLK digital filter is used both by the EAC and by channel 0 to get full synchronization between the two logics.

The eTPU Angle Counter (EAC) logic runs continuously and updates the TCR2 Angle counter, eliminating the microcode latency in updating the TCR2 value.

37.15.6.3 STAC interface

Both time bases TCR1 and TCR2 can be shared between the Engines and with other blocks in the same MCU. Each one of both eTPU engines can drive their time bases to the STAC (Shared Time and Count) bus, acting as a server, while any other block can capture the value into its resources and behave like a client. For further reference about the STAC bus operation see [eTPU Engine 1 STAC Configuration Register \(eTPU_TBR_STAC_ENG1\)](#) and [eTPU Engine 2 STAC Configuration Register \(eTPU_TBR_STAC_ENG2\)](#).

The eTPU can export to the STAC bus or import from the STAC bus the following internal resources:

- TCR1: Can be exported to or imported from the STAC bus. TCR1 can only be imported from STAC bus when the engine is not in Angle Mode. When TCR1 is imported from the STAC bus, it becomes read-only for the microcode and reflects the imported values. For details, see [Timer Count Register 1 - TCR1](#).
- TCR2: Can be exported to or imported from the STAC bus. TCR2 can only be imported from the STAC bus when engine is not in Angle Mode. When TCR2 is imported from the STAC bus, it becomes read-only for the microcode, and reflects the imported values. When exported to the STAC bus, TCR2 can work in either Angle Mode or as a free running counter associated with the TCRCLK signal. For details, see [EAC - eTPU Angle Counter](#).

Proper configuration of the following bits is necessary to determine what can drive the STAC bus: eTPU_TBR_TBCR_ENGn[AM] and eTPU_TBR_STAC_ENGn[REN2, RSC2], according to the following table.

Table 37-356. STAC bus and host read sources

AM (eTPU_TBR_TBCR_ENGn)	REN2, RSC2 (eTPU_TBR_STAC_ENGn)	TCR2 bus source (Host read of eTPU_TBR_TB2R_ENGn)	STAC bus driver
00	0x (disabled)	TCR2/Time	×
01, 10 or 11	0x (disabled)	TCR2/Angle	×
00	11 (Server)	TCR2/Time	TCR2/Time
01, 10 or 11	11 (Server)	TCR2/Angle	TCR2/Angle
01, 10 or 11	10 (Client)	not allowed ¹	

1. STAC client configuration in Angle Mode is also not allowed for TCR1.

Note that Angle Mode is not available for STAC bus clients: configuring both at the same time brings unspecified results and must be avoided. When TCR2 is a stand-alone counter or a STAC Bus server, the same value that is driven to the internal TCR2 bus is also exported to the STAC bus (either Time Count or Angle).

STAC bus configuration is provided by the eTPU_TBR_STAC_ENGn fields REN1/2 and RSC1/2. REN1/2 enable the STAC interface to interact with the resource (either TCR1 or TCR2 bus). RSC1/2 configure the resource (either TCR1 or TCR2 bus) as Server or Client.

Each time base / angle count resource from each engine receives a unique 4-bit hard-wired address that identifies it as a potential server. This address is used by the STAC Controller to coordinate which resource will drive the bus at a given STAC time-slot. For any time-slot there is a server driving the bus upon selection of the STAC Controller, and there may be a client linked to that server by the eTPU_TBR_STAC_ENGn fields bits SRV1/2 on each Engine. When the server address on the Red Line STAC bus matches

the value in SRV1/2, the client will load the STAC information into the appropriate resource. For information on eTPU STAC Bus configuration, see [eTPU Engine 1 STAC Configuration Register \(eTPU_TBR_STAC_ENG1\)](#) and [eTPU Engine 2 STAC Configuration Register \(eTPU_TBR_STAC_ENG2\)](#).

The eTPU does not include a STAC Controller module, which is instantiated once in the system integration.

Note

Setting a timebase as client of itself is not allowed.

37.15.6.4 GTBE - Global Time Base Enable

[GTBE](#) enables time bases in both engines, allowing them to be started synchronously. GTBE is divided in two block interface signals: the eTPU GTBE output and the eTPU GTBE input. GTBE bit sets the eTPU GTBE output, and the eTPU GTBE input enables time bases to start. The eTPU GTBE output signal can be used by MCU integration for synchronization between eTPU time bases and time bases from other modules. If [GTBE](#) must enable only the eTPU time bases, the eTPU GTBE output is simply connected to the eTPU GTBE input. These two cases are shown in the figure below. Synchronization logic can be as simple as an OR logic gate.

Once the eTPU GTBE input transitions to 1, the Engine 1 Time Bases start 1 eTPU clock earlier than Time Bases in Engine 2, except when TCRCLK is selected as clock source or TCR1 when eTPU_TBR_TBCR_ENG n [TCR1CS]=1. This happens independently of prescaler values as long as they are the same for both engines, because the prescalers also freeze when the eTPU GTBE input = 0. Microcode can always write to TCR1/2 registers, with either value of the eTPU GTBE input.

NOTE

The timebase prescalers are reset when the GTBE input is negated.

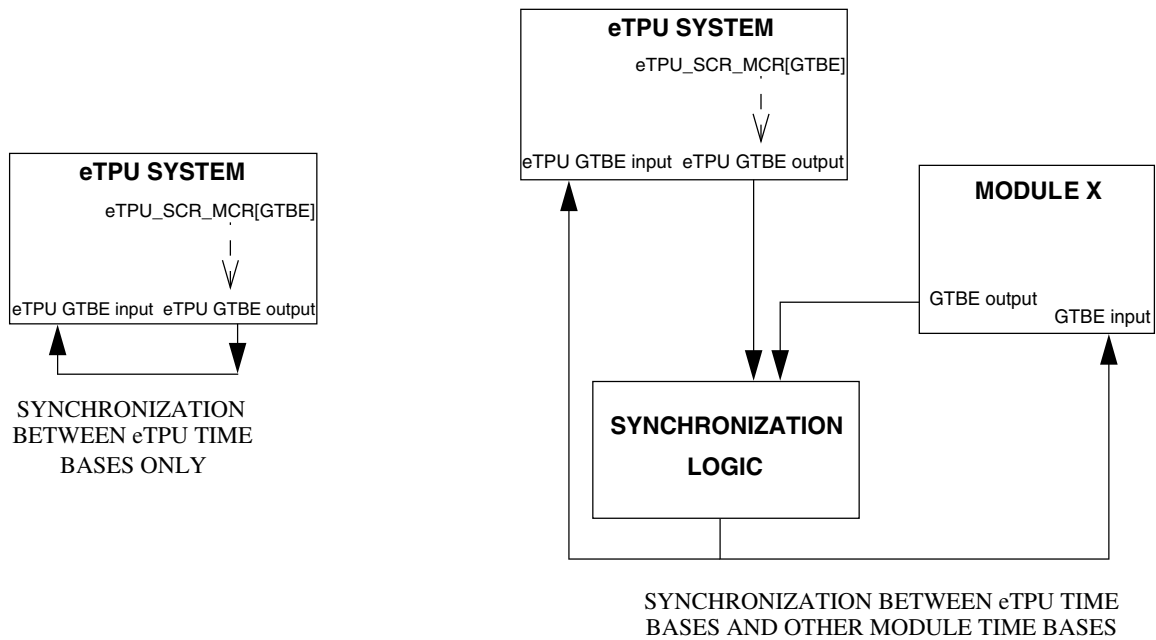


Figure 37-319. Time base synchronization

37.15.6.5 TCRCLK digital filter

The TCRCLK signal has an improved integrating digital filter with a 2-bit up-down counter. The counter counts up to 3 when a high signal level is detected, or down to 0 when a low level is detected. The signal state is updated to one when the counter stops at 3, or zero when the counter stops at 0. The `eTPU_TBR_TBCR_ENGn[TCRCF]` field (see [eTPU Engine 1 Time Base Configuration Register \(eTPU_TBR_TBCR_ENG1\)](#) and [eTPU Engine 2 Time Base Configuration Register \(eTPU_TBR_TBCR_ENG2\)](#)) determines whether the TCRCLK signal input (after a synchronizer) is filtered with the same filter clock as the channel input signals (see [Enhanced Digital Filter - EDF](#)) or uses the eTPU clock divided by 2, and also whether the TCRCLK digital filter works in integrator mode or the same two sample mode as the channel filters (see [Table 37-339](#)).

The TCRCLK filter delay and prescaling determines the minimum detectable TCRCLK pulse widths and, therefore, its maximum frequency, as shown in [Filter clock prescaler](#) and [Table 37-355](#). The TCRCLK signal delay from the module input to TCR1/TCR2 incrementing or detection in the EAC logic is explained in [Input/Output Signal Delays](#).

37.15.7 EAC - eTPU Angle Counter

37.15.7.1 General

The EAC logic contains a mechanism which follows the flywheel angle, based on the tooth rate. This hardware works in combination with the TCRCLK signal, the TCR2 counter and Channel 0, 1 or 2 (depending on the eTPU_TBR_TBCR_ENG n [AM] field) to generate angle information on the TCR2 bus which is passed to all the local engine channels. The EAC helps to implement a digital angle PLL (see [Figure 37-323](#)), which combines hardware with microcode processing at channel 0, 1 or 2. The angle measurement is based on history knowledge of the tooth period, for predicting the period of the next tooth. The tooth period is partitioned into a programmable number of **Angle Ticks**. The eTPU application will use the divider in the MAC/Divide unit to calculate an integer and a fraction part of the angle tick such that the full tooth period gets the correct programmed number of angle ticks with no accumulated error.

Each single tooth can be divided in angle ticks, up to 1024. In a 60 teeth flywheel, 128 Angle Ticks per tooth provide resolution of ~0.05 degrees per tick, which meets the accuracy requirement of 0.1 degrees in current automotive applications.

The measurement of one tooth in angle ticks is independent on engine RPM; it is the tooth period itself (and the corresponding tick period) that is re-calculated for each new tooth, based on the difference between the estimated tooth and the actual detection.

For these applications, one of the eTPU channels 0, 1 or 2 is dedicated to service the physical tooth detection. Channel 0 shares the same filtered input as the TCRCLK signal to get the same timing as the EAC. The TCRCLK edge detection is selected by the eTPU_TBR_TBCR_ENG n [TCR2CTL] field for the EAC, and by IPACA/B on channel 0, which must be set to detect the same edge(s). When channels 1 or 2 are selected to work with the EAC, IPACA/B is used to select the tooth signal edge detection for both the channel and the EAC, and the tooth signal that feeds the EAC is the same filtered input which feeds the channel.

Channel 0, 1 or 2 generates the signal transition service request, and can also be used for generation of a window filter on this transition, to qualify TCR2 clocks. For this purpose, the selected channel should be configured with double match window filtering mode (refer to [Channel Modes](#)). Depending on the channel mode set for the channel, Match A recognition opens the window, and Match B recognition may close it or leave it open. See [Angle Logic and Channel Modes](#) for details. Match B also generates a time-out service request. Its input signal transition comes from the tooth. The window can be defined by microcode to open at a predefined point inside the tooth period, and stay open for a desired percentage to the tooth period. The window can be measured in angle or time. This method improves the noise immunity by allowing transition detection only on an expected period, a feature which was software responsibility in previous TPU versions.

The EAC supports deceleration, acceleration, last tooth and missing tooth scenarios. The large range of angle ticks per tooth can be used to cover longer tick counts caused by one or more missing teeth, or to provide extra resolution for future application requirements. In case of a missing tooth, the EAC can be configured to insert a dummy tooth or to simply measure a longer tooth.

Figure 37-324 shows the block diagram of the Angle Counter system. TCR1 or TCR2 prescaler is used as a time base which measures the tooth period and is used for partitioning the period to angle ticks.

37.15.7.2 Angle Mode registers

In Angle Mode, the registers described below control eTPU angle operations. They are accessible only by microengine as source and destination registers in microinstructions. When eTPU is not in angle mode (eTPU_TBR_TBCR_ENGn[AM] field is negated), all Angle Mode registers can be used as general purpose registers.

37.15.7.2.1 TPR - Tooth Program Register

TPR provides configuration for the Angle Counter circuit. In this register, the microcode can properly adjust the tooth count (controlling last tooth, missing teeth, dummy tooth insertion, halt until tooth detection) and the number of angle ticks per tooth (field TICKS). Note that this register is sampled into a temporary register in the EAC logic when the High Rate Mode is detected (see High Rate Mode (Acceleration) and TPR Buffering), which means that changes to this register may take effect only for the next tooth.

See Count Control and High Rate Logic and to Special Cases of Missing Teeth and Last Tooth until Handling False Tooth Detection for a detailed explanation about the use of this register. The figure below provides a detailed description of TPR register.

Several conflict issues on TPR writes are explained in Special TPR write cases.

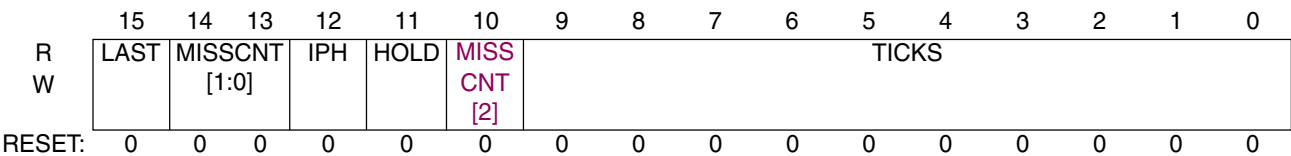


Figure 37-320. TPR

LAST —Last Tooth Indication
Asserted by microcode and negated when a tooth is detected or inserted via IPH.

Table continues on the next page...

1 = Last Tooth - reset TCR2 Counter at the end of the tooth tick count (after physical tooth or IPH=1) when MISSCNT=0.

0 = Not Last Tooth.

If the tooth is detected or inserted before the tooth tick count completes (going High Rate mode, see [High Rate Mode \(Acceleration\)](#)), LAST resets immediately, but TCR2 resets only when the tooth count completes and MISSCNT=0.

MISSCNT — Missing Tooth Counter

Decrement on each estimated tooth, stops at zero. Used for generation of up to 7 "Dummy Teeth" whenever it holds a non-zero value.

If the tooth is detected or inserted before the missing tooth tick count completes (going High Rate mode, see [High Rate Mode \(Acceleration\)](#)), MISSCNT resets immediately, but missing teeth count continues in High Rate mode (see [TPR Buffering](#)).

IPH — Insert Physical Tooth

This bit generates a dummy physical tooth which has the same effect as a real physical tooth, and resets itself subsequently. If EAC is in Halt mode, it switches back to Normal mode¹. If EAC is in Normal Mode, it switches to High Rate Mode. If Angle Logic is frozen by HOLD=1 (see below), it returns to the state it was at the freezing moment.

1 = Insert dummy physical tooth.

0 = No Operation.

Note: IPH reads as 1 in the next microinstruction after it is asserted, negating subsequently. However, it can be set twice in two consecutive microinstructions to generate two teeth and make the EAC go from Halt to Normal to High Rate Mode.

HOLD — Force EAC Hold

This bit forces the EAC to halt its operation in a special EAC freeze mode until a new physical tooth (a real one or emulated with IPH=1) is detected. Assertion of this bit immediately freezes the EAC in the middle of the tooth period. When a new physical tooth is detected, the bit is automatically negated by the EAC. The HOLD bit can be used for synchronizing the EAC tooth count, in case that a false physical tooth is detected due to noise.

1 = Force EAC to halt until detection of a physical tooth.

0 = Normal Operation.

TICKS — Angle Ticks Number in the Current Tooth

This field defines the number of angle ticks in the current physical tooth. It partitions the tooth period to the required number of angle ticks. The actual number of angle ticks in a tooth is (TICKS+1).

Note: In High Rate mode (see [High Rate Mode \(Acceleration\)](#)), TPR writes are immediately effective only for bits IPH and HOLD. All other fields changes are "buffered" and become effective when EAC leaves High Rate mode. See also [Special TPR write cases](#).

Note: Bits LAST, IPH and HOLD must not be asserted all at once

1. Missing a physical tooth naturally causes EAC to get into Halt mode.

37.15.7.2.2 TCR2 - Timer Counter 2

In Angle Mode, TCR2 (see the following figure) counts angle ticks instead of time.

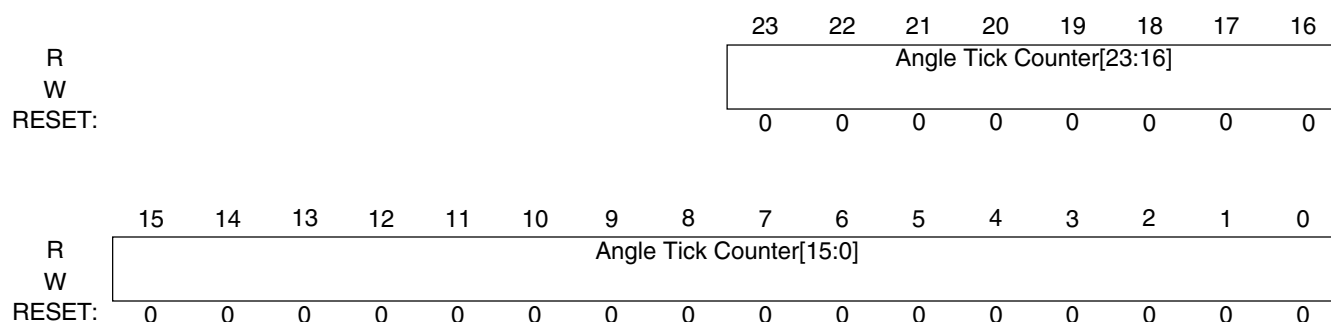


Figure 37-321. TCR2 in Angle Mode

This 24-bit free-running counter is used to generate an accumulated Angle Fraction value. It is updated by the Angle Tick Generator - refer to [Angle Tick Generator](#) for more details. Refer to [Count Control and High Rate Logic](#) for a detailed explanation about the use of this register in Angle Mode.

TCR2 provides continuous count of the angle in units of angle ticks. The Angle Tick Counter in TCR2 can be reset due to "Last Tooth" microcode indication. TCR2 Prescaling is disregarded in Angle Mode: physical tooth detection is done by EAC regardless of the value set in TCR2P.

37.15.7.2.3 TRR - Tick Rate Register

The exact period of the Angle Tick is programmed in the Tick Rate Register (see the figure below) by microcode. The TRR time unit depends on eTPU_TBR_TBCR_ENG n [ATGC] (see [Figure 37-324](#)):

- if ATGC=0, the TRR time unit is the period of TCR1 clocks taken as eTPU clocks divided by $2 \cdot (\text{TCR1P} + 1)$, even if TCR1CS=1 (see [eTPU Engine 1 Time Base Configuration Register \(eTPU_TBR_TBCR_ENG1\)](#) and [eTPU Engine 2 Time Base Configuration Register \(eTPU_TBR_TBCR_ENG2\)](#)).
- if ATGC=1, the TRR time unit is given by the eTPU clock divided by $2 \cdot (\text{TCR2P} + 1)$, using the TCR2 prescaler fed by eTPU clock/2 (independently of TCR1CS).

See [Calculating the Angle Tick Period Integer and Fraction](#) for a complete description about the mechanism to calculate the value to be written into this register.

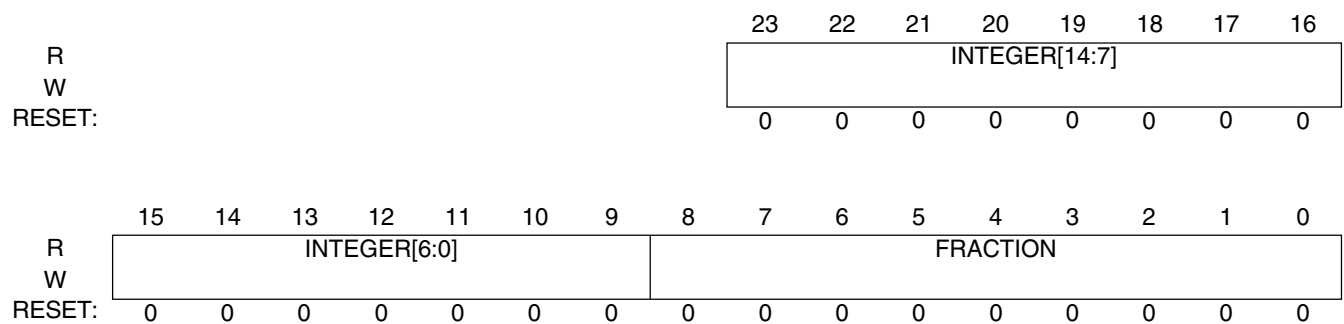


Figure 37-322. TRR

INTEGER — The integer part of TCR1/TCR2 clocks in one Angle Tick.

This number, decremented by one, works as a down-counter preload value. A value of **INTEGER**=0 represents an integer of 32768. A new value written is reloaded into the counter (becoming effective) when a new tick starts or a tooth is detected or inserted via IPH.

FRACTION — Nine-bit fractional part of TCR1/TCR2 clocks in one angle tick.

The **FRACTION** value is accumulated in the EAC Fraction Accumulator, and whenever the result overflows (i.e., the accumulated fraction added up to an integer), the Tick Prescaler is halted for one TRR time unit.

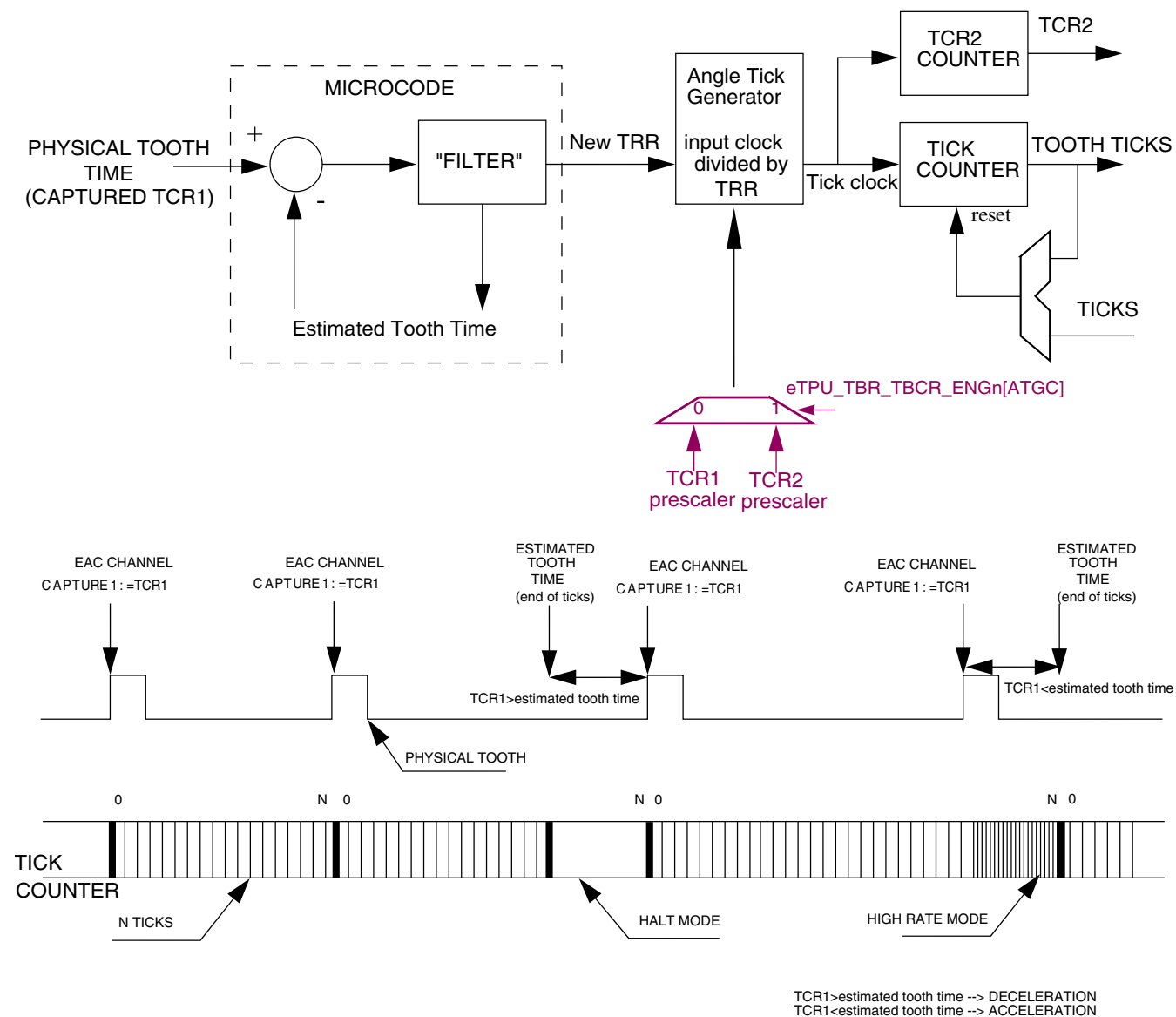


Figure 37-323. EAC "PLL"



Figure 37-324. eTPU angle counter system

37.15.7.3 Acceleration and Deceleration

Acceleration and deceleration affect the new tooth period relative to the known period of the last tooth. Changes in tooth period may be extreme at very low engine RPM (such as cold start and warm start). The worst case of tooth period changes is caused during missing teeth, since there is more time for changes in angular velocity to be unnoticed by the EAC hardware. For example, on cold start (~20 RPM) there may be extreme acceleration: the ratio between a known tooth period before two missing teeth and the new tooth period after the missing teeth can be very high (up to a factor of 75). Acceleration and deceleration effects from tooth to tooth are less extreme as the engine climbs to high RPM.

In case of deceleration, the estimated tooth period ends before the actual tooth detection arrives. In this case, the EAC hardware waits at the end of the current tooth period, when it is said to be in **Halt mode**, until the real tooth indication is received, then continues with normal operation (**Normal mode**). See [Figure 37-323](#).

In case of acceleration, the actual tooth period is shorter than the estimated tooth period. As a result, a new physical tooth indication arrives before the end of the estimated tooth period. In this case the EAC closes the gap on **High Rate mode** by counting on eTPU clock divided by eight to the end of the tooth, advances to the next tooth, and switches back to normal operation mode. See [Figure 37-323](#).

The reason that the EAC does not jump directly to the next tooth is the need to provide sequential angle count throughout the whole tooth period, for channels or external STAC clients (if TCR2 is a STAC server) which compare angle in "equal" mode. These peripherals must get all the valid angle values in a sequential manner, to avoid missing angle matches.

TCR2 advancing from one tooth to another is a continuous count, and can be optionally reset at the end of the tooth. An estimated tooth is generated after the Tooth Tick Counter reaches the TICKS programmed value.

The EAC works continuously and switches automatically between Normal, Halt and High Rate modes. It relies on the microcode to calculate the estimated tooth period on every tooth, and to update the correct angle tick and tooth parameters in the EAC control registers. On high RPM, tooth period changes are reduced from tooth to tooth, and the EAC may follow the angle with good accuracy for several teeth without microcode intervention.

The EAC handles missing teeth by insertion of "dummy" teeth, or by enlarging the expected tooth period. It is a good practice to locate the flywheel missing teeth in non-critical angles, since a missing tooth may increase the angle measurement error (acceleration and deceleration is detected late).

37.15.7.4 Angle Tick Generator

The Angle Tick Generator is responsible for generating a programmed number of angle ticks in the tooth period. It generates the ticks in an average rate which ensures completion of the correct number of angle counts in the estimated period of the tooth, since the count of one tooth in angle ticks is independent on engine RPM. The main output of the Angle Tick Generator is the tick clock that feeds TCR2 in Angle Mode, as well as the internal Tooth Tick Counter (see [Figure 37-324](#)). The Tooth Tick Counter counts ticks within a tooth, from 0 up to TICKS, is controlled by the Angle Tick Generator logic and cannot be accessed by microcode. Refer to [Figure 37-325](#) for a generic presentation of the angle tick count and the measurement of a single tooth period.

37.15.7.4.1 Calculating the Angle Tick Period Integer and Fraction

On each tooth the microcode has to update the exact period of a single angle tick used for counting the internal angle in the tooth. The period of an angle tick is determined by the EAC configuration, see [Generating the angle ticks](#) for details.

The microcode can use the eTPU MAC Divider unit (see [MAC and Divide Unit \(MDU\)](#)) to divide the tooth period by the number of angle ticks per tooth, which is stored in the TICKS field of TPR (refer to [Angle Mode registers](#)). This division yields the integer part of the angle tick period and the remainder. Dividing again the remainder shifted left nine positions, by the number of angle ticks per tooth translates the remainder to a 9-bit fraction. The microcode concatenates the 15-bit integer and the 9-bit fraction to a 24-bit value and writes it to TRR. The new rate is effective immediately after the next Angle Tick is generated by the Angle Tick Generator³¹.

For high RPM, note that shifting the tooth period value nine positions to the left prior to the first divide operation would calculate, in one operation, the integer and the fraction. For example: On 60 teeth flywheel running at 1000 RPM, tooth period is 1 msec. If TCR1/TCR2 counts @25 MHz, it counts 25,000 times in a tooth, which can be represented by 15 bits. Therefore the tooth period can be shifted nine positions to the left prior to divide operation, and be represented with 24 bits.

Using shift left nine positions and one divide operation would get the result in MACL register (in MDU) which holds the integer and nine bits of the fraction:

```
Angle_Tick_Rate {Integer[14:0], Fraction[8:0]} = (ToothPeriod <<9) / Ticks
TRR = Angle_Tick_Rate {Integer[14:0], Fraction[8:0]}
```

31. In High-rate mode, the tick keeps being updated at the rate of eTPU clock/8 until it goes back to Normal mode, when the new TRR value is used.

NOTE

In the pseudocode above, the tooth period (ToothPeriod) is not, in general, the value of estimated tooth time. It is obtained by microcode by subtracting TCR1 values between two teeth detections, converted to TCR2 periods if ATGC=1. Its comparison with the estimated tooth time indicates acceleration (if minor) or deceleration (if greater) to the microcode.

On low RPM the initial tooth period, measured in TCR1/TCR2 counts, may be too big to be shifted nine positions to the left. For lower RPM (for example 500 RPM) the tooth period cannot be represented by 15 bits, and shifting it nine positions to the left would lose the MSB. In this case, two divide operations are required as follows: first divide the Tooth Period by the number of TICKS - the integer is stored in MACL and remainder in MACH. MACL is saved in another register. MACH is shifted 9 positions to the left and divided again by TICKS. In parallel with the second divide, the register which saved the original MACL is shifted left 9 positions. After the divide MACL contains the 9 bits fraction and the other register contains the 15-bit integer, shifted left nine times. The logical OR of the two registers is written to the TRR:

```
Angle_Tick_Rate {Integer[14:0]} = (ToothPeriod) / Ticks
Remainder[9:0] = (ToothPeriod) mod Ticks
Angle_Tick_Rate {Fraction[8:0]} = (Remainder[9:0] << 9) / Ticks
TRR = Angle_Tick_Rate {Integer[14:0], Fraction[8:0]} = (Integer[14:0] << 9) | Fraction[8:0]
```

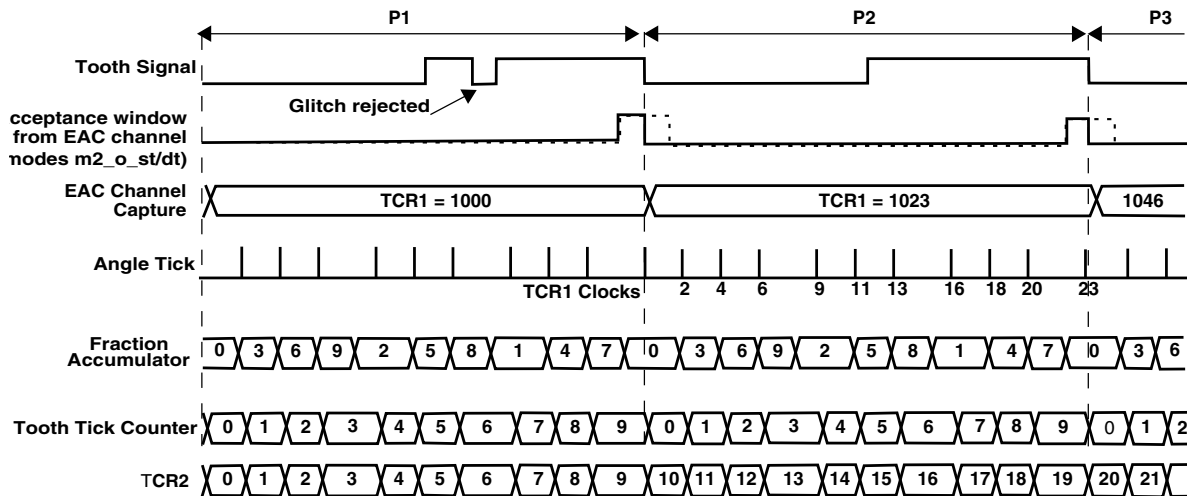


Figure 37-325. Angle Ticks Generation

37.15.7.4.2 Generating the angle ticks

The integer part of TRR is preloaded to a prescaler, which counts in time intervals as follows (see [Figure 37-324](#)):

- If $\text{eTPU\_TBR\_TBCR\_ENGn[ATGC]}=0$, one unit of TCR1 clock period (if $\text{TCR1CS}=0$) or twice the TCR1 clock period (if $\text{TCR1CS}=1$)³², that is, $\text{eTPU clock period} \times 2 \times (\text{TCR1P}+1)$ (independently of TCR1CS).
- If $\text{ATGC}=1$, two units of the eTPU clock period multiplied by the TCR2 prescaler factor, that is, $\text{eTPU clock period} \times 2 \times (\text{TCR2P}+1)$.

When the down counter reaches zero, it generates an angle tick pulse to the Angle Counter Logic and a Load pulse to the Fraction Accumulator. It is then preloaded with most updated TRR integer part. Due to the Load pulse, the 9-bit fraction is accumulated in a 9-bit Fraction Accumulator. If a fraction overflow condition occurs (the 9-bit adder asserts carry out), the accumulator saves the lower 9 bits of the addition result, which is the remaining fractional part. The carry out bit indicates an accumulated integer "one" which means that the angle tick is early by one input clock. It halts the prescaler operation for one input clock to compensate the accumulated error generated by the integer prescaler. As a result, the average angle tick period takes into account both the integer and the fraction parts. The accuracy depends on the bit count of the fraction. Using 9-bit fraction part while the width of the field TICKS in register TPR is 10 bits provides accuracy of two LSB on a full scale ($\text{TICKS}=1023$) or one LSB on lower scale ($\text{TICKS} \leq 511$).

When the Tick Prescaler gets High Rate mode indication from the Angle Counter Logic, it generates angle ticks at a rate of eTPU clock divided by eight. In this case it does not generate Load pulses to the Fraction Accumulator, ignores its "hold" input and preloads internally to a fixed period of eight eTPU clocks. When High Rate mode is entered, the prescaler is preloaded to a period of eight eTPU clocks before its first angle tick generation, ensuring separation of at least eight eTPU clocks between the last Normal mode angle tick and the first High Rate mode angle tick. The fraction accumulator resets when the tick count advances to the next tooth, or when TRR is written by the microcode.

37.15.7.5 Count Control and High Rate Logic

The Count Control and High Rate Logic controls TCR2 operation in Angle Mode, using the angle ticks generated by the Angle Tick Generator. Count Control logic is responsible for advancing, holding and resetting the TCR2 and Tooth Tick Counter in the proper timing, such that the TCR2 time base will reflect the correct estimated angle. This logic also includes the Tooth Program Register (TPR, see [Angle Mode registers](#) for more information).

32. The EAC logic overall runs at a rate of eTPU clock divided by 2, therefore the tick generator takes the halved TCR1 clock when $\text{TCR1CS}=1$ and $\text{ATGC}=0$.

The Count Control and High Rate Logic handles deceleration, acceleration, missing teeth and last tooth. On High Rate (acceleration) it ensures that the angle bus scans all valid angle values in a rate which can be traced by the STAC bus. This operation enables external STAC clients (if TCR2 is a STAC server) or channels working in "equal-only" comparator mode to match the TCR2 exported angle information in "equal" mode, in an exact match.

Because the eTPU channels are capable of capturing either TCR1 or TCR2 due to signal transition, the microcode can get either the angle or time of the physical pin transition. Since the EAC channel (0, 1 or 2) is connected to the physical tooth, the microcode can get the EAC error in angle domain (tooth appears at the wrong angle) or time domain (physical tooth captured time into the EAC channel, relative to the estimated tooth time). Note that in angle mode, the transition detect logic of the channel 0, if selected as the EAC channel, is fed from the digital filter of the TCRCLK signal, and not from the channel 0 internal digital filter. This ensures synchronous operation of channel 0 and the EAC hardware.

Another feature of the eTPU channel, when working in Single Match And Single Transition Enhanced Mode (refer to [Single Match Enhanced Mode \(sm_st_e\)](#)), is capturing a single time base due to signal transition before and after the digital filter. This option allows subtracting the digital filter delay to get accurate signal transition timing on the channel. This way, the TCRCLK signal may be programmed with a slow and reliable digital filter, and get accurate time measurement of the digital filter delay.

To assert the end of the estimated tooth period the Count Control and High Rate logic compares the TICKS field in TPR (refer to [Angle Mode registers](#)) with the current value of the tooth tick counter. When the Tooth Tick value is greater or equal to TICKS, it determines the end of the estimated tooth period. On acceleration this event occurs during High Rate mode operation, after the arrival of a physical tooth. In deceleration, this event occurs during Normal Mode, before the arrival of a physical tooth. On constant angular velocity, this event appears together with the arrival of a physical tooth.

The following sections describe the operation of the Counter Control and High Rate logic.

37.15.7.5.1 Normal Mode

In Normal mode the Counter Control logic advances TCR2 and the Tooth Tick counter as if the engine has a constant speed during the tooth period. It receives the angle ticks from the Angle Tick Generator in an average rate which is determined by the Tooth Rate Register (TRR). This is the reset mode.

When the Tooth Tick Counter is about to reach the last value effectively stored in TPR field TICKS plus one, the hardware detects the end of the estimated tooth period. If the physical tooth and the estimated tooth arrive at the same time the EAC stays in Normal mode, the Tooth Tick Counter is reset, and TCR2 is incremented (depending on TPR bits LAST and MISSCNT). If the physical tooth and the estimated tooth do not arrive at the same time, either acceleration or deceleration is detected, and the EAC switches to the proper mode. See the figure below for a detailed diagram of Normal Mode behavior.

The microcode which services the EAC channel physical tooth transition may update TRR according to various conditions to give the best estimation of the current tooth period, according to the previous tooth period and other engine parameters.

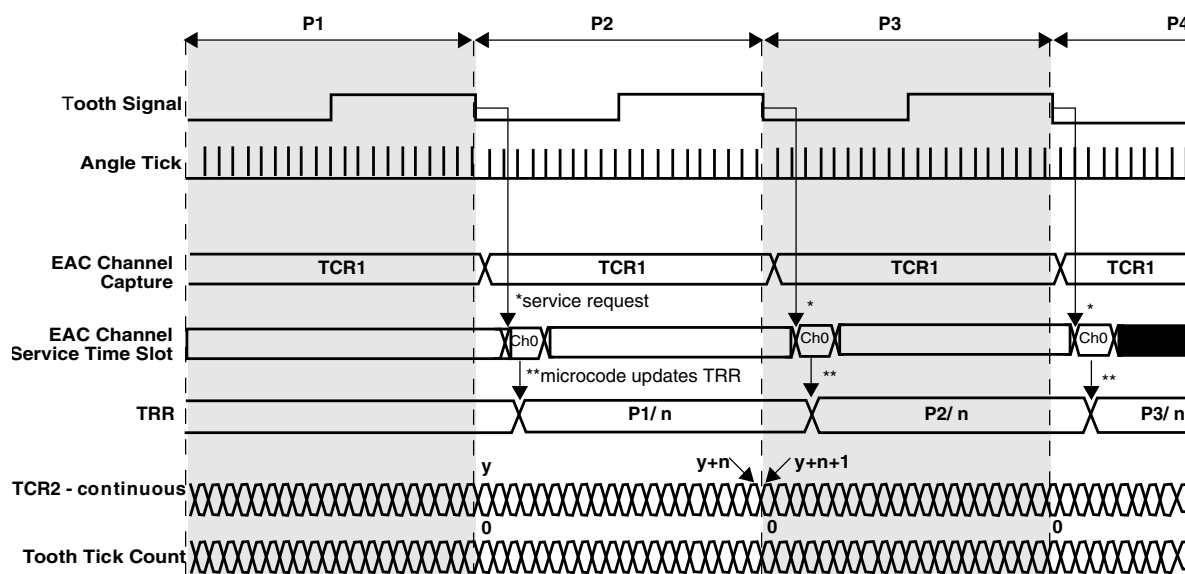


Figure 37-326. Normal Mode

37.15.7.5.2 Halt Mode (Deceleration)

In case of deceleration, the Tooth Tick Counter reaches the TICKS value before the arrival of the next tooth. The Count Control logic does not reset neither advances TCR2 and Tooth Tick Counters. The Count Control logic halts TCR2 and Tooth Tick Counters at the end of the tooth, waiting for the physical tooth to arrive.

When the physical tooth is detected the EAC switches back to Normal Mode and releases TCR2 to count the angle ticks of the new tooth, also resetting the Tooth Tick Counter. Only then TCR2 may wrap to 0, if TPR bit LAST is asserted. See the figure below for a detailed diagram of Halt Mode behavior.

The microcode service caused by the physical tooth determines the deceleration, calculates the new tooth period and Angle Tick period and updates TRR. This operation slows the angle tick rate generated by the Angle Tick Generator on-the-fly, to the rate required for the new tooth period.

Since the microcode service is initiated by the physical tooth edge, microcode latency may introduce a small angle error caused by using the TRR value of the previous tooth at the beginning of the current tooth. On high RPM, deceleration is relatively small but the microcode latency may take a significant percentage of the tooth period. On low RPM microcode service latency takes little percentage of the tooth period, but there may be cases of extreme acceleration and deceleration. The microcode latency can be calculated knowing TCR1 value during the service time, and TCR1 value captured in the EAC channel due to the physical tooth pin transition. The duration of the Halt mode is obtained using the estimated tooth time.

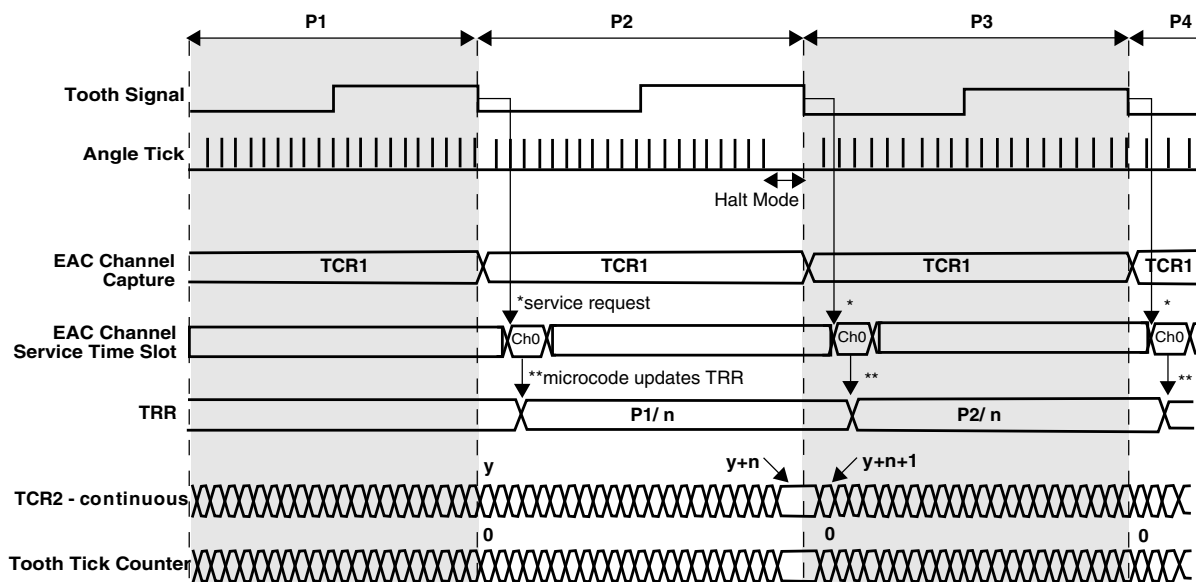


Figure 37-327. Halt Mode - Deceleration

37.15.7.5.3 High Rate Mode (Acceleration)

In case of acceleration, the next tooth arrives before the Tooth Tick Counter reaches the TICKS value. In this case the High Rate logic is responsible for closing the gap and advancing the tooth on the correct timing. The High Rate mode operates as follows:

- When the acceleration is detected (physical tooth arrives before the Tooth Tick Counter reaches the TICKS value), the Count Control and High Rate logic switches to High Rate mode in which both the Tooth Tick Counter and TCR2 count at rate of

eTPU clock divided by eight, until the Tooth Tick Counter reaches the current TICKS value. To ensure correct operation, the TICKS value is sampled in the logic at the beginning of the mode.

- At this point, which represents the estimated tooth edge, the logic resets the Tooth Tick Counter and advances TCR2 (or resets it if LAST is asserted and MISSCNT = 0).
- The control logic switches back to Normal Mode, using the most updated TRR value as input to the Angle Tick Generator. The logic samples the updated TICKS value for the tooth estimation, last tooth indication and number of missing teeth from TPR.

In High Rate mode the angle ticks are provided at high speed until the end of the current tooth. This operation is required to scan all the valid angle values of the current tooth, in a rate which is not too high for the STAC bus continuous update, but much higher than the rate dictated by TRR.

EAC channel microcode, which services the physical tooth transition detection, can start its service either before High Rate mode operation is complete (the Tooth Tick Counter has not reached the TICKS value) or after the EAC switched back to Normal mode. Any physical teeth received while the EAC is in High Rate Mode does not alter the immediate EAC state, but it is still detected by the EAC channel logic and can, therefore, alter future EAC behavior (for instance, closing the tooth detection window - see [Angle Logic and Channel Modes](#)).

At the beginning of High Rate mode operation, the TPR value is preloaded into a temporary register in the Counter Control Logic, used for scanning all the valid values to the end of the current tooth, with its appropriate LAST and MISSCNT attributes. While the EAC is in High Rate mode operation, the effect of microcode update of TPR fields LAST, MISSCNT and TICKS is delayed to the next estimated tooth, after the High Rate mode operation is complete (see [Special TPR write cases](#)). This is because the current physical tooth represents the next estimated tooth. If the microcode updates this field after High Rate mode operation is complete, the current physical tooth and estimated tooth are the same, and the effect is immediate. Either in High-Rate mode or not, the value read by microengine is the same written, even if not yet effective, until the EAC resets LAST and/or IPH, or decrements MISSCNT. Typically the microcode service may occur during the High Rate mode on extreme acceleration situation at low RPM. Therefore, the microcode operations are always related to the real physical tooth. From the above it can be seen that the microcode updates of the TICKS field in TPR affect the end time of the current physical tooth. For correct operation, this field should be updated before the Tooth Tick Counter has reached either the old or the new TICKS value.

During High Rate mode operation, TRR is ignored and the Angle Tick Generator uses eTPU clock divided by eight. Therefore, the TRR update by microcode will take effect only after the EAC switches back to Normal mode. If microcode service occurs after the Tooth Tick Counter has been reset, the EAC is already back in Normal mode, and some angle ticks may have been counted at the rate of the previous tooth. In this case the new TRR value will have immediate effect on the angle tick period, and the microcode should take into consideration the delay from the physical tooth to the estimated tooth in calculation of the next tooth period. See the figure below for a detailed diagram of High Rate Mode behavior.

An angle error may be introduced by the duration of the High Rate mode. Also, the scheduler latency may introduce a small accumulated error by using TRR value of the previous estimated tooth at the beginning of the current tooth. After the estimated tooth has advanced, the duration of the High Rate mode operation is the actual delay from the physical tooth edge to the estimated tooth edge. This delay can be obtained by comparing the estimated tooth time with the EAC channel capture register which captured TCR1 on the physical pin transition.

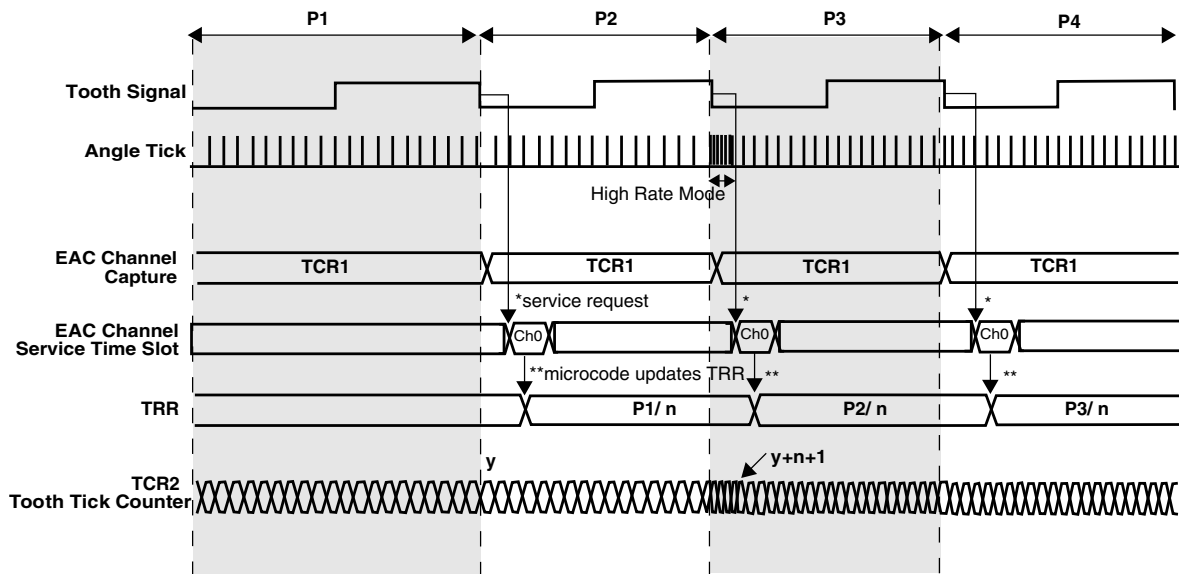


Figure 37-328. High Rate Mode - Acceleration

37.15.7.6 Special Cases of Missing Teeth and Last Tooth

The EAC handles cases of up to three missing teeth and the last tooth in the engine cycle. The following paragraphs describe these functions.

37.15.7.6.1 Handling the Last Tooth

The microcode can set the TCR2 counter to work in engine periods (wrap-around count) or continuous angle measurement.

For periodic operation, during the last engine cycle tooth the EAC microcode has to set the LAST flag in TPR. As a result, when the tooth period is ended, the Counter Control Logic generates a reset command to both the Tooth Tick Counter and TCR2 instead of an advance command. The operation resets the TCR2 based angle count, indicating a new period of the engine cycle. This implementation provides an engine cycle based periodic angle measurement.

37.15.7.6.2 Handling Missing Teeth

The EAC can handle up to three missing teeth in two ways:

- Count the angle ticks relative to the last physical tooth. The microcode should update the TPR TICKS field to the number of angle ticks included in two, three or four teeth, according to the flywheel type (one, two or three missing teeth). EAC hardware works in its regular manner.
- Insert a "dummy" tooth instead of the missing tooth, at the estimated point in time. After the "dummy" tooth, the Angle Tick Counter is incremented as if there was a physical tooth. A "dummy" tooth can be inserted only during Normal or High Rate operation modes. The microcode inserts "dummy" teeth by writing to the MISSCNT field in TPR.

In the first option the missing tooth is not counted on the angle measurement. For example, a flywheel with 59 physical teeth and one missing tooth can be considered as 58 identical teeth numbered (0-57) and tooth number 58 has a double number of angle TICKS. In this case a 720 degrees engine cycle has 118 teeth. TCR2 reflects the real angle, since it counts angle ticks continuously.

In the second option, the missing teeth are counted as "regular" teeth by automatic insertion of "dummy" teeth. The microcode has to write a non-zero value to the MISSCNT field in TPR. This field is a 3-bit down counter which affects the operation of the Counter Control logic.

For example, a flywheel with 59 physical teeth (0-58) and one missing tooth (59) can be considered as 60 teeth numbered (0-59), all having the same number of angle ticks. The microcode has to write "001" to the MISSCNT bits during the period of tooth number 58 to indicate that next tooth (59) is missing.

When the Tooth Tick Counter reaches the TICKS value, TCR2 is incremented as if a physical tooth has been detected. In addition, the MISSCNT value initializes a "dummy tooth counter" which is decremented to indicate the number of left "dummy teeth" which still need to be generated. Because a dummy tooth was counted, EAC does not enter Halt Mode and Tooth Tick Counter continues incrementing in the absence of a physical tooth detection.

In case of extreme acceleration on very low RPM (cold start) there can be a situation that the first physical tooth after one or two missing teeth appears even before the "dummy" tooth is generated. Due to the acceleration the EAC switches to High Rate mode in order to run through all the valid angle values, including the dummy teeth. When the Tooth Tick Counter reaches the TICKS value on High Rate mode, and the "dummy tooth" down counter is not zero, the generated "dummy tooth" advances to the next tooth and decrements the "dummy tooth" counter, but does not switch the EAC back to Normal mode. The last "dummy tooth" decrements the counter to zero, indicating that no more dummy teeth are to be inserted, and the next tooth is an estimated physical tooth. The EAC continues at High Rate mode until the Tooth Tick Counter reaches the TICKS value again, then advances to the next tooth while switching back to Normal mode. When in High Rate mode, the TPR register does not reflect the MISSCNT downcounting, see [TPR Buffering](#) for details.

MISSCNT can be rewritten before it reaches 0, allowing it to count more than 7 missing teeth, as long as no physical tooth arrives between the first MISSCNT write and the rewrite.

37.15.7.6.3 Combining Missing Teeth and Last Tooth

The Last Tooth indication takes effect when there are no more missing teeth to be generated, i.e the "dummy tooth" counter value is zero. If, for example, the microcode sets the missing teeth counter to "10" (two missing teeth) and sets the LAST flag, the first and the second dummy teeth will increment TCR2, and the third estimated tooth, which correlates with the physical tooth (the first of the next cycle), will reset TCR2, because LAST was set. This scheme enables the microcode to define one or more missing teeth to be replaced by "dummy tooth" insertion, and the end of the engine cycle in one service request. It is assumed that the two missing teeth must come together in the same engine cycle, and not split between two engine cycles (either the missing teeth are both last in an engine cycle or both not last, but not last in one engine cycle and first in the next).

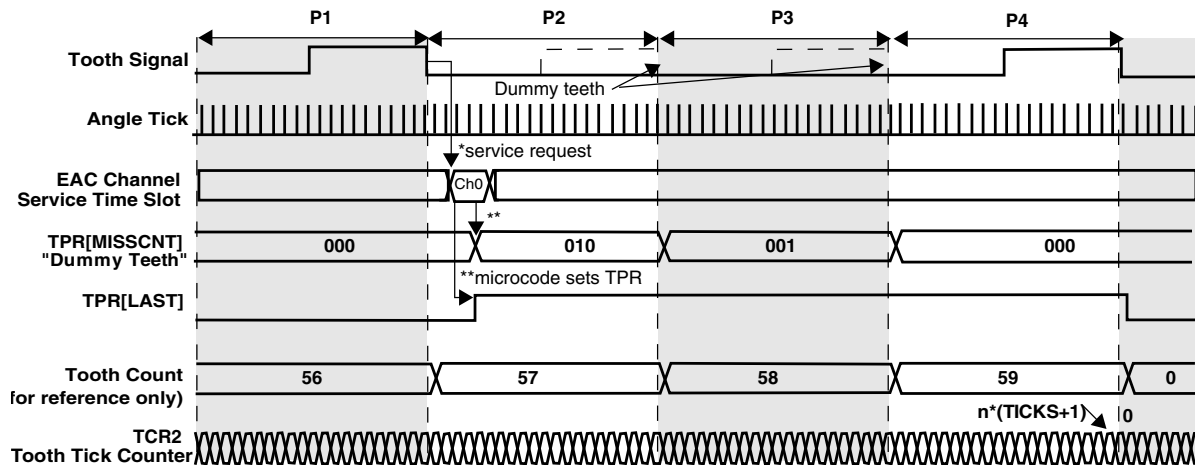


Figure 37-329. Missing Teeth and Last Tooth Combination

37.15.7.7 Handling Mechanical Tooth Correction

The EAC can handle tooth edge detection errors caused by flywheel mechanical errors. The eTPU application can hold a vector of tooth mechanical errors with one entry per tooth. This error can be measured in angle ticks which are independent of engine RPM. The TRR can be updated to the fixed period of any tooth, including its mechanical error.

Because TCR2 counts continuously, without being reset, the mechanical correction is transparent. Though the tooth has its own programmed TICKS value, TCR2 simply counts angle ticks, disregarding the boundary between two adjacent teeth.

37.15.7.8 Handling Mis-detected Tooth

When a physical tooth signal is missed by the engine sensor, the EAC may get into Halt mode at the end of the estimated tooth period, expecting the physical arrival. In this case, a Match timeout event of EAC channel will call service which detects extreme deceleration. The microcode can assert the IPH bit in TPR, to force the detection of the missed physical tooth. It can also calculate the accumulated angle bus error, and fix the next estimated tooth period, to close the gap.

37.15.7.9 Handling False Tooth Detection

Most of the false tooth detection, caused by noises on the engine tooth sensor, can be eliminated by the window blanking filtering, timed by EAC channel match recognitions. The EAC also provides means of fixing false detection of an additional tooth which passed the window filter. When such an event occurs, the EAC switched to High Rate

mode (advancing to the next tooth) and when the next physical tooth arrives, an extreme acceleration is detected: the EAC sees the remaining portion of the current tooth period as another tooth period. The microcode can detect the situation when the acceleration is not realistic, or when immediately after the detection of this extreme acceleration, the following tooth indicates extreme deceleration back to the original RPM.

When the microcode detects such a case, the Tooth Counter has been advanced by mistake to the next tooth. The microcode can set the HOLD bit in the TPR, forcing the EAC to freeze and wait for the next physical tooth to close the gap. When the next physical tooth arrives, HOLD is automatically negated and the EAC proceeds from that point to the remaining portion of the tooth period, in the same mode it was when HOLD bit was asserted.

37.15.7.10 Angle Logic and Channel Modes

The TCRCLK transition detection is qualified by a signal that comes from channel 0 (see [Figure 37-324](#)) and depends on the particular mode (PDCM) programmed for that channel. This configures a window for TCRCLK detection for the angle logic which is the same (except on High Rate mode, see [High Rate Mode \(Acceleration\)](#)) used to set the TDLA flag on single transition modes, and TDLB on double transition modes (see signals TSE1, TSE2 in [Figure 37-307](#)). The same applies when channels 1 or 2 are used to control EAC (see signals TS1, TS2 in [Figure 37-307](#)). As a consequence, the window depends on the channel mode as follows:

- on all modes, the window closes upon a tooth edge detection: TDLA asserted on single transition modes, TDLB asserted on double transition modes.
- on mode m2_st: the window opens on Match A (which enables Transition A) and does not close with Match B. If Match B comes before Match A, it blocks Match A and, hence, Transition A.
- on mode m2_o_st: the window opens on Match A (which enables Transition A) and closes on Match B. Match B is enabled by Match A, so it cannot come before.
- on all other Single Transition modes, the window is "always open", independently of matches.
- on mode m2_dt: the window opens on Transition A, which is enabled by Match A. The window does not close with Match B, but if it comes before Match A the later gets blocked and, hence, blocks Transitions. Match A is also a condition for the window, so the microcode closes it by clearing MRLA.

- on mode m2_o_dt: the window opens on Transition A, which is enabled by Match A. The window closes on Match B, which is enabled by Match A. Match A is also a condition for the window, so the microcode closes it by clearing MRLA.
- on all other Double Transition modes, the window opens on Transition A.

37.15.7.11 Restarting angle logic

It is not recommended to change the eTPU_TBR_TBCR_ENGn[AM] field while GTBE=1. However, if the angle logic must be restarted without interfering with the timebase count running on TCR1, the procedure below must be followed:

1. Program eTPU_TBR_TBCR_ENGn[AM]=00 and TCR2CTL=111 at once. That prevents TCR2 from incrementing while the angle logic is disabled. The angle logic state machine resets to Normal mode and the tick prescaler to the initial count with AM=00, but not the microengine registers TPR and TRR.
2. Start a thread to reconfigure the EAC. The thread must set the EAC controlling channel (0, 1 or 2) flags in a state, depending on the channel mode, that lets the channel tooth detection window open (see [Angle Logic and Channel Modes](#)). It can optionally write TCR2 with an angle preset value equivalent to the first tooth expected after restart. The thread must also set TPR bit HOLD=1. The TPR bit IPH must be 0.
3. After the thread is finished, program eTPU_TBR_TBCR_ENGn[AM]=01, 10 or 11, and TCR2CTL according to the desired tooth edge selection if AM=01.

The first tooth detected after this procedure restarts the TCR2 counting, unfreezing the Angle Mode logic into normal mode.

37.15.7.12 Special TPR write cases

This section describes how simultaneous modification of TPR fields are resolved, and how the effect of TPR writes depend on the EAC mode.

37.15.7.12.1 TPR Buffering

In High Rate mode (see [High Rate Mode \(Acceleration\)](#)), TPR writes are immediately effective only for bits IPH and HOLD. Writes to all other fields are "buffered" and become effective when EAC leaves High Rate mode. However, if TPR is written a

second time right after IPH is asserted in Normal mode, this second write behaves as if EAC is still in Normal mode. Only in the next microcycle (after execution of a nop, for instance) the TPR writes are buffered, acknowledging High Rate mode.

MISSCNT and LAST can be written any value during High Rate mode, and the value that prevails for the next tooth is the one sampled when EAC goes back to Normal mode (or the value written in Normal or Halt mode thereafter). If MISSCNT and/or LAST are not zero when High Rate mode begins, they are sampled into the internal EAC logic and are effective while High Rate lasts (missing teeth count continues, and TCR2 is reset at the end of High Rate if LAST=1). However, their values in TPR are reset when High Rate mode starts. After that and until the end of High Rate mode, the value read by microcode is the same written. This behavior prevents read-modify-writes to TPR from unwillingly rewriting LAST or MISSCNT.

37.15.7.12.2 IPH and LAST

If both IPH and LAST are asserted in the same microinstruction with HOLD=0, the EAC acts as if LAST was set first and then IPH right after, so that:

- in Normal mode, it goes to High Rate with LAST=1;
- in Halt mode, it goes to Normal Mode resetting LAST (and TCR2);
- in High Rate mode, IPH is ignored and LAST becomes effective in the next tooth (physical or inserted) after it goes back to Normal mode.

37.15.7.12.3 IPH and TICKS

Because of different results depending on the EAC mode at the time of TPR write, it is not advisable to write 1 to IPH and change TICKS at the same microinstruction. A consistent behavior is obtained if IPH is written first and TICKS on the second microinstruction after (for instance with a nop between them), making the new TICKS value valid for the next tooth regardless of the mode.

The mode-dependent behavior is (assuming HOLD=0):

- in Normal mode, the new TICKS value becomes valid before EAC goes High Rate due to the IPH;
- in Halt mode, the EAC goes to Normal mode, and new TICKS is valid for the next tooth;
- in High Rate mode, the new TICKS value is effective when EAC leaves High Rate mode, and IPH is ignored;

37.15.7.12.4 IPH and MISSCNT

If both IPH and MISSCNT are written non-zero values with HOLD=0:

- in Normal mode, at the next microcycle the EAC goes to High Rate mode, the MISSCNT field in TPR goes to 0, and the missing teeth are counted in High Rate mode.
- in Halt mode, the EAC goes to Normal mode for one microcycle and then, yet another microcycle later, goes to High Rate mode, counting the missing teeth. The TPR fields IPH and MISSCNT are zeroed on the transition from Normal to High Rate mode.
- in High Rate mode, IPH is ignored (resetting at the next microcycle) and MISSCNT is buffered (see [TPR Buffering](#)).

37.15.7.12.5 IPH and HOLD

If IPH and HOLD are asserted at once, IPH cancels the HOLD and both reset. The EAC is not frozen, regardless of the EAC mode (High Rate, Normal or Halt).

37.15.7.12.6 LAST and HOLD

If LAST and HOLD are written 1 at once, LAST asserts and EAC is frozen. Afterwards, when a physical tooth is detected or IPH is asserted, the EAC is unfrozen in the same state it was before, and LAST is kept asserted.

37.15.8 Microengine

Each eTPU engine has a microengine that fetches, decodes and executes microinstructions. The Microengine only works when there are service requests to be attended, otherwise it turns to idle state, controlled by Hardware Scheduler (see [Scheduler](#)).

Microcode is stored in Shared Code Memory (SCM) that is 32-bit wide. Microengine can access SDM using a different bus from the one used to access code memory, so that code and data can be accessed at the same time (Harvard Architecture).

Some of eTPU functionality can only be made through the microengine, like configuring channels and interrupting host. Microengine gives eTPU a high degree of flexibility, since any desired treatment for channel's events can be implemented; however, that

flexibility comes at the cost of channel service's latency. Latency is worsened when channels from a same eTPU engine contend for microengine service. In the figure below, a block diagram of microengine architecture is shown.

Follows a summary of Microengine features:

- P, DIOB, A, B, C, D, SR, RAR, LINK, CHAN, MACL, MACH, ERTA, ERTB, TCR1, TCR2, TPR, TRR registers are accessible by microcode.
- 24-bit ALU and Post-ALU shifter performs basic arithmetic and logical operations described in [ALU and Post-ALU Shifter](#).
- MDU (MAC/Divide Unit) performs integer MAC, multiply and divide operations.
- Fixed Microinstruction Size of 32bits.
- Fixed-length instruction execution (2 eTPU clocks)
- Static superscalar operation

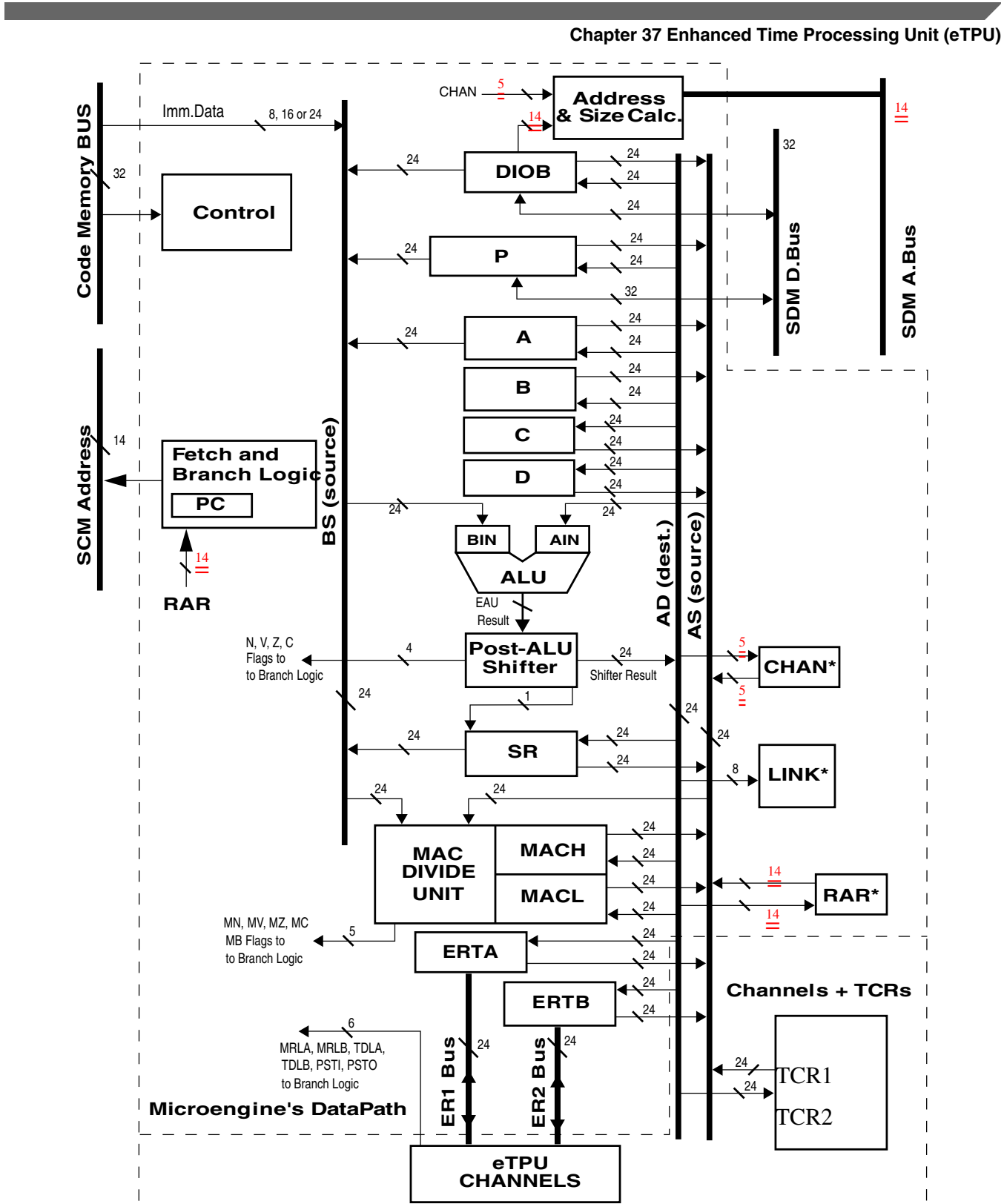


Figure 37-330. Microengine Block Diagram

37.15.8.1 Registers

eTPU microengine accesses a total of 18 registers. Fourteen of them are special purpose (registers A, B, C and D are for general use). Special purpose registers except CHAN and LINK can also be used as general use if the operation that use their contents are not performed. Register description is intended to just introduce their functionalities and not to provide detailed explanation of it since it will be described in [Microinstruction Set](#). Registers less than 24 bits in size are right-justified.

None of the registers have guaranteed reset values. However, some are initialized just before the thread starts (see [Time Slot Transition](#)).

37.15.8.1.1 P Register

P register is the only one that is 32-bit wide. It can be used as source and destination for arithmetic/logical operation, and as source and destination for SDM read/write operations.

For P source/destination possibilities in ALU/MDU microoperations, see [Selecting sources and destination](#).

When P is used as SDM read/write operations source or destination there are only 3 possibilities of access: all 32 bits, lower 24 bits and upper 8 bits. SDM operations are explained in detail in [SDM Microoperations](#).

P is automatically loaded with one parameter before the thread starts (parameter preload). For more information see [Entry Point format](#) and [Time Slot Transition](#).

Upper 8 bits of P register can be used as application state, since these bits can be tested as branch conditions. P[31:24] is also used in dispatch microoperation (see [Dispatch Microoperation](#)), and bit pairs P[29:28], P[27:26], P[25:24] can be directly copied into Channel flags 1 and 0 using field FLC. Together with Entry Table Condition Encoding, it provides fast state resolution without code execution.

37.15.8.1.2 DIOB Register

DIOB is an abbreviation for Data Input/Output Buffer. DIOB is 24-bit wide and can be used as source and destination for arithmetic/logical operations as well as SDM data source and destination. DIOB only can be accessed as 24 bits, both in arithmetic/logical and SDM read/write operations. When using DIOB to perform an SDM access, only lower 24 bits of SDM will be accessible (SDM upper 8 bits always remains unchanged).

DIOB can also be used as SDM addressing register, when DIOB contents is used as absolute SDM address (14-bit wide). In this case DIOB can also be pre-decremented or post-incremented (see [Indirect Addressing Mode](#)).

DIOB is automatically loaded with one parameter before the thread starts (parameter preload). For more information see [Entry Point format](#) and [Time Slot Transition](#).

37.15.8.1.3 ERTA and ERTB Registers

ERTA/B registers are 24-bit wide and can be used as source or destination in arithmetic/logical operations. ERTA/B are the only source for channel's match registers write (see [Write Channel Match and UDCM Registers](#)). ERTA can also be the source for UDCM write.

When a thread starts to be executed, ERTA and ERTB are loaded with a copy of CaptureA and CaptureB registers respectively. ERTA/B can be used to receive a copy of MatchA and MatchB registers. ERTA/B are the only destination of MatchA/B read operation (see [Special T4ABS Source Operation: Read Match Registers](#)).

ERTA and ERTB also receive a copy of CaptureA and CaptureB registers when CHAN register is written (see [CHAN Register](#)). For more information about Capture and Match registers see [MatchA and MatchB Registers](#) and [CaptureA and CaptureB Registers](#).

37.15.8.1.4 SR register - Shift Register

SR is a 24-bit wide register that can be used as source and destination register for arithmetic/logical operations. SR can shift right its contents by 1 bit at time and, at the same time, receive in its bit 23 the lost bit of a shift-right operation in post-ALU shifter ([ALU and Post-ALU Shifter](#)), allowing SR to be used to perform 48-bit shift right (see [Shift Operations](#)).

37.15.8.1.5 MACH and MACL Registers

Both MACH and MACL are 24-bit registers, part of MAC/Divide unit (see [MAC and Divide Unit \(MDU\)](#)). They can be used as source and destination in most arithmetic/logic operations. When multiply or divide operations are used (multiply-accumulate included), MACH and MACL have special purpose and some restrictions apply, see [MAC and Divide Unit \(MDU\)](#) for more information.

37.15.8.1.6 LINK Register

Link Register is an 8-bit wide register and can be used only as destination in arithmetic operations. LINK is a write-only command register, which precludes its use as a source register for ALU operations. When LINK register is written, it issues a service request for the channel number and eTPU engine equal to the number written in LINK register (see [Functions and threads](#) and [Channel Link](#) for information about Link Service Request).

37.15.8.1.7 RAR Register

RAR is a 14-bit register and can be used as source and destination in arithmetic operations. RAR also receives the contents of PC register when a subroutine call is executed. The contents of RAR register is loaded into PC when a return from subroutine is executed. RAR is loaded with value 0x3FFF during TST. For more information about subroutine call and return see [Branch Operations](#) and [Return from subroutine](#) respectively.

37.15.8.1.8 CHAN Register

CHAN is a 5-bit register that can be used as source and destination in arithmetic operations. The contents of CHAN register affects the execution of many channel-related microinstructions, because its number indicates the **selected channel**. CHAN register must not be used to store temporary values in arithmetic operations. For more details, refer to [Channel Selection Register - CHAN](#).

37.15.8.1.9 Counter Registers: TCR1, TCR2, TPR and TRR

All these registers are 24-bit wide except TPR, which is a 16-bit register. They can be read or written in arithmetic/logical operations, and have special-purpose uses for time base and angle mode operations. For more information about those registers see [Time bases](#) and [EAC - eTPU Angle Counter](#).

37.15.8.1.10 General Purpose Registers: A, B, C and D

A, B, C and D are 24-bit general purpose registers, which can be used to store intermediate values and do not have other specific uses with any eTPU feature.

37.15.8.2 ALU and Post-ALU Shifter

The ALU executes 24-bit arithmetic and logical operations. ALU's output goes directly to a 1-bit shifter, called post-ALU shifter, so it is possible, for example, to add and shift using only one microinstruction.

In some microinstruction formats, it is not possible to specify the operation executed by ALU. In these cases ALU will always perform addition.

In formats which have the field ALUOP for ALU operation selection, all of them can be performed, including add/subtract using C (carry) flag as ALU's carry-in, bitwise AND/OR/NOT/XOR, and shift/rotate of 2, 4, 8 and 16 bits. See [ALU/MDU Operation Selection](#).

Subtraction, inversion, increment and decrement can be performed by combinations of source inversion and setting ALU's carry-in to 1.

ALU always performs 24-bit operations on its inputs, called **A-source** and **B-source**, and outputs a 24-bit result. 8-, and 16-bit inputs are zero padded to 24 bits. Likewise, ALU 24-bit output is always truncated to the destination register size.

37.15.8.2.1 ALU flags

Four flags - Carry, Negative, Overflow, Zero - described below, are related to ALU and post-ALU shift operations. Operation size and shifting affect flags generation logic. Operation size determines the result boundary to be used for flags generation. Operation size is determined by size of sources and destination (see [Flags Sampling Control](#)). For more information about flag generation, see [Flags Sampling Control](#). ALU flags can be used as branch condition (see [Conditional/unconditional branch](#)) or conditional ALU/MDU operation (see [Conditional ALU/MDU Operation Execution](#)).

Field CCS/CCSV in microinstructions can force **no update** of all flags. Not all flags are updated in all ALU operations: Overflow is updated only on addition and absolute value operations, Carry flag is updated in most ALU operations, and only Zero and Negative are updated in all ALU operations.

ALU flags are never updated when microinstruction starts an MDU operation, regardless of CCS/CCSV, but are updated normally afterwards, on ALU operations that are executed in parallel with an ongoing MDU operation (MDU has its own flags).

Note

Operation size can be smaller than destination register. For example: 0xFFFF + 0x0001 (both 16-bit sources) stores 0x10000 in a 24-bit register and sets Zero and Carry flags because operation size is 16 bits.

37.15.8.2.1.1 Carry flag (C)

In an unsigned addition without shifting, Carry flag is the ALU carry from bit 7 to 8, 15 to 16, or 23 to 24 on 8-, 16- and 24-bit operation sizes respectively. In an unsigned subtraction without shifting, Carry Flag represent the sign of ALU's result considering operation size (Carry Flag equal to 1 means a negative result).

Carry flag definition is operation-dependent. The Carry flag in add/subtraction with Post-ALU shift is defined in [Table 37-362](#) . Find the definitions for other operations in the following sections.

37.15.8.2.1.2 Negative flag (N)

Negative flag indicates the sign of result based on the operation size, regardless of the operation performed, as shown in the following table.

Table 37-358. Negative (N) flag behavior

Operation size	Value
8 bits	$N = \text{result}[7]$
16 bits	$N = \text{result}[15]$
24 bits	$N = \text{result}[23]$

Note

The N flag may not reflect the sign of the value actually written into the destination register, if it does not have the same size of the operation (see [Flags Sampling Control](#)). This is always the case for registers RAR (14 bits) and CHAN (5 bits).

37.15.8.2.1.3 Overflow (V)

Overflow is updated only on addition (with or without carry) and absolute value operations. In signed operations, overflow flag indicates that the result of arithmetic operation (add or subtraction) can not be represented by a word of the size of the operation. Overflow flag behavior for addition is defined in the following table. Overflow flag for Absolute operation is explained in [Absolute Value operation](#) . V Flag is calculated using ALU adder output (i.e., it is not affected by 1-bit shift/rotate operations).

Table 37-359. Overflow flag on addition - V

Op. size	Value ¹
8 bits	$(AS[7] \& BS[7] \& !\text{alu\_adder\_output}[7]) \vee (!AS[7] \& !BS[7] \& \text{alu\_adder\_output}[7])$
16 bits	$(AS[15] \& BS[15] \& !\text{alu\_adder\_output}[15]) \vee (!AS[15] \& !BS[15] \& \text{alu\_adder\_output}[15])$
24 bits	$(AS[23] \& BS[23] \& !\text{alu\_adder\_output}[23]) \vee (!AS[23] \& !BS[23] \& \text{alu\_adder\_output}[23])$

1. BS is taken after any inversion by the BINV field, but not added to the carry bit (CIN field)

NOTE

For V-flag definition on the absolute operation, see “Absolute Value Operation”.

37.15.8.2.1.4 Zero flag (Z)

Zero flag equal to 1 indicates that the result from the ALU, limited to the operation size, is zero, regardless of the operation performed, whether the result is written, or where it is written. It depends on the operation size, as shown in the following table.

Table 37-360. Zero flag - Z

Operation size	Value
8 bits	$Z = (\text{result}[7:0] == 0x00)$
16 bits	$Z = (\text{result}[15:0] == 0x0000)$
24 bits	$Z = (\text{result}[23:0] == 0x000000)$

37.15.8.2.2 ALU ADD operation with and without shifting

ADD operation is selected by ALUOP or ALUOPI fields and when none of them is available in a microinstruction format³³. Optionally, result can be shifted or rotated right by 1 bit, which is selected by SHF, ALUOP or ALUOPI fields. See [Microinstruction Set](#) for more details. The following table describes how CIN and BINV fields change ADD operation behavior.

Table 37-361. Types of ADD operations

BINV	CIN	Operation (adder output)
1	1	AS + BS
1	0	AS + BS + 1
0	0	AS - BS
0	1	AS - BS - 1

ALU adder output can be 1-bit shifted or 1-bit rotated right as follows:

Shift right:

```

if BINV==1
    result [23:0] = adder_output [24:1]
else
    result [23:0] = adder_output [24:1] xor 0x800000
endif

```

Shift left:

33. ALU operations only occur on formats where a destination field is found (T2ABD/T2D).

Functional description

```
result[23:1] = adder_output[22:0]
result[0]     = 0
```

Rotate right:

```
case (opsize/CCSV)
8-bit:
    result[6:0] = adder_output[7:1]
    result[7]   = adder_output[0]
    result[23:8] = adder_output[23:8]
16-bit:
    result[14:0] = adder_output[15:1]
    result[15]   = adder_output[0]
    result[23:16] = adder_output[23:16]
24-bit:
    result[22:0] = adder_output[23:1]
    result[23]   = adder_output[0]
```

Note

Only for the Post-ALU rotate right, the operation size is determined by the field CCSV (see [Flags Sampling Control](#)). For example: if CCSV=00, T4ABS=P (24-bits), T4BBS=A (24 bits), T2ABD=B (24 bits), and ALUOP="Add ROR", then B gets A+P with bits 7:0 rotated, even though the operation size is 24 bits.

The following table describes Carry flag behavior.

Table 37-362. Carry flag update on ADD operation

BINV ¹	Op. size	shift/rotate	Value
1	8 bits	none	adder carry from bit 7 to bit 8
1	16 bits	none	adder carry from bit 15 to bit 16
1	24 bits	none	alu_adder_output[24]
0	8 bits	none	!adder carry from bit 7 to bit 8
0	16 bits	none	!adder carry from bit 15 to bit 16
0	24 bits	none	!alu_adder_output[24]
0 or 1	8 bits	shift left	alu_adder_output[7]
0 or 1	16 bits	shift left	alu_adder_output[15]
0 or 1	24 bits	shift left	alu_adder_output[23]
0 or 1	x	shift right	alu_adder_output[0]
1	8 bits	rotate right	adder carry from bit 7 to bit 8
1	16 bits	rotate right	adder carry from bit 15 to bit 16
1	24 bits	rotate right	alu_adder_output[24]
0	8 bits	rotate right	!adder carry from bit 7 to bit 8
0	16 bits	rotate right	!adder carry from bit 15 to bit 16
0	24 bits	rotate right	!alu_adder_output[24]
except on max-constant generation (see Max Const Generation With T4BBS=111)			

1. BINV has no effect on carry-out when used to code max constant (see [Max Const Generation With T4BBS=111](#)).

Flags N and Z on shift are updated according to the result after shift. Flag V with shift is updated according to the ADD operation only, the same way as without shift.

37.15.8.2.3 ADC Operation

ADC operation is selected by the ALUOP field. CIN field is ignored when this operation is selected. The following table describes how BINV change ADC operation behavior.

Table 37-363. Types of ADC operations

BINV	CIN	Operation
1	x	AS + BS + C flag
0	x	AS - BS - C flag

Flags behave exactly the same way as for ADD operation without shift/rotate.

37.15.8.2.4 Bitwise Operations

Bitwise AND, OR and XOR are selected by ALUOP field. On these operations CIN field is ignored and BINV field inverts (bitwise NOT) BS. C and V Flags are never updated on these operations. The following table describes AND, OR and XOR bitwise operations.

Table 37-364. Types of Bitwise Operations

ALUOP	BINV	Operation
10000	1	AS BS
10000	0	AS (~BS)
10001	1	AS ^ BS
10001	0	AS ^ (~BS)
10010	1	AS & BS
10010	0	AS & (~BS)

37.15.8.2.5 Set Bit / Clear Bit Operations

These operations set or clear the AS bit determined by BS[4:0]. If the bit number resolves to a value greater than 23, no bit is set or cleared (i.e., result is equal to AS). On these operations CIN field is ignored and BINV field inverts (bitwise NOT) BS. C and V flags are never updated for set/clear bit operations. These operations **override** B-Source size to 8 bits.

set bit (BINV = 1):

Functional description

```
result = AS | (1 << BS)
```

clear bit (BINV = 1):

```
result = AS & ~(1 << BS)
```

set bit (BINV = 0):

```
result = AS | (1 << (31 - BS))
```

clear bit (BINV = 0):

```
result = AS & ~(1 << (31 - BS))
```

37.15.8.2.6 Exchange Bit

Exchange the AS bit determined by BS with C flag. If the bit number resolves to a value greater than 23, no exchange is performed (i.e., result is equal to AS and C flag is not updated). This operation **overrides** BS size to 8 bits. On this operation, CIN field is ignored and BINV field inverts (bitwise NOT) BS. V flag is never updated on exchange bit operation. C flag is always updated, regardless of CCSV, unless BS > 23.

Exchange Bit (BINV=1):

```
if BS <= 23
begin
    temp_C_flag = AS[BS]
    if C_flag == 1
        result = AS | (1 << BS)
    else
        result = AS & ~(1 << BS)
    C_flag = temp_C_flag
end
```

Exchange Bit (BINV = 0):

```
if (31 - BS) <= 23
begin
    temp_C_flag = AS[31 - BS]
    if C_flag == 1
        result = AS | (1 << (31 - BS))
    else
        result = AS & ~(1 << (31 - BS))
    C_flag = temp_C_flag
end
```

37.15.8.2.7 Multibit shift/rotate operations

These operations shift or rotate AS by 2, 4, 8 or 16 bits. Size of shift/rotate is determined by BS[1:0]. The following table describes the number of shifted/rotated bits depending on BS[1:0] value.

Table 37-365. Number of shifted/rotated bits for each BS[1:0] value

BS[1:0]	Bits shifted/rotated
0	2
1	4
2	8
3	16

Shift right is a logical operation (i.e., zeros are inserted on left). Multibit shift and rotate operations **overrides** BS size to 8 bits. The shifts and rotate operate on 24 bits, independently of the operation size.

V flag is never updated for multibit shift or rotate operations. Carry flag behavior is described in the following table. CIN is ignored in these operations, but BINV is effective.

Table 37-366. Carry flag value on multibit shift/rotate operations

ALUOP	BS[1:0]	C flag value ¹
11001 (shift left)	0	AS[22]
11001 (shift left)	1	AS[20]
11001 (shift left)	2	AS[16]
11001 (shift left)	3	AS[8]
11010 (shift right)	0	AS[1]
11010 (shift right)	1	AS[3]
11010 (shift right)	2	AS[7]
11010 (shift right)	3	AS[15]
11011 (rotate right)	0	AS[2]
11011 (rotate right)	1	AS[4]
11011 (rotate right)	2	AS[8]
11011 (rotate right)	3	AS[16]

1. CCS/CCSV can disable flag update on multibit shift/rotate, but the specified flag size in CCSV is ignored for the C flag.

37.15.8.2.8 Absolute Value operation

Absolute Value operation is selected by ALUOP field. On this operation, AS is interpreted as a signed number and its absolute value is the result. V and N flags are updated with the result signal determined by the operation size. AS bit 23 after size override and sign extension (if any, see [A-Source Size Override](#)), regardless of A-source register size, is used to check the operand signal and is copied to C-flag. Note that if AS is 8-bit or 16-bit, its sign is taken into account and copied to C only if sign-extension is performed. The following table summarizes flag updating for Absolute Value operation.

This operation is independent of B-source. Instruction fields T4BBS, BINV and CINV are ignored in this operation. The Absolute Value operation size is the minor between A-source size and Destination size.

Table 37-367. ALU flags in Absolute Value operation

operation size	V, N ¹	C	Z
8	alu_output[[7]	AS[23]	alu_output[7:0] == 0
16	alu_output[15]		alu_output[15:0] == 0
23	alu_output[23]		alu_output[23:0] == 0

1. V, N can be 1 on 8- and 16-bit Absolute Value, because the operand sign is always taken from bit 23. V, N can also be 1 in 23-bit Absolute Value (or 8-bit and 16-bit with sign extension), if the operand is 0x800000 (0x80, 0x8000).

37.15.8.3 MAC and Divide Unit (MDU)

MDU is an autonomous resource in the microengine which can carry out sequential multiply, multiply-accumulate, fractional multiplication and divide operations, selected through the microinstruction fields ALUOP or ALUOPI. The unit supports signed and unsigned multiply and fractional multiplication of any combination of 8, 16 or 24 bit operands ³⁴, and also signed and unsigned 24-bit multiply-accumulate. Divide operation is unsigned, and both operands are always 24-bit wide.

Depending on the size of operands and the type of operation, MDU can take more than one microcycle to execute the operation, but microengine continues to execute microinstructions in parallel. When the microcode issues an END command, any MDU executing operation terminate immediately and is left incomplete. When selecting an operation that uses MDU, the result is always placed in MACH and MACL registers, and the register selected as destination does not have its value changed ([Selecting sources and destination](#)). During calculations, MACH and MACL holds temporary values and should not be written, otherwise the result is unspecified. One must not start an MDU operation while MDU is already busy: the result is unspecified for both the ongoing operation and the started one.

MDU Operations update its own set of 5 flags, described in [MDU Flags](#). MDU operations never update C, N, V and Z flags. CIN and BINV microinstruction fields affect MDU operations according to the following table.

34. There is no distinct selection of 24-bit fractional multiplication, for it works exactly as a 24-bit ordinary multiplication.

Table 37-368. CIN and BINV with MDU Operations

B-source operand	BINV	CIN	Op. Performed
signed	1	1	AS mdu_op BS
	0	0	AS mdu_op (-BS)
	1	0	reserved
	0	1	reserved
unsigned ¹	1	1	AS mdu_op BS
	1	0	AS mdu_op (BS+1)
	0	x	reserved

1. includes the B-source (unsigned) in fmults (signed) operations.

37.15.8.3.1 Multiply and Multiply-Accumulate Operation Length

MDU needs 2 sources, A source and B source, to perform an operation. The time needed to perform a multiply or multiply-accumulate is:

- on 24-bit x 8 bit multiplies: 2 microcycles (one start-MDU plus one execution microcycle)
- on 24-bit x 16 bit multiplies: 3 microcycles (one start-MDU plus two execution microcycles)
- on 24 bit x 24 bit multiplies/macs: 4 microcycles (one start-MDU plus three execution microcycles)

An internal pipeline in MDU allows multiply-accumulate (or even non-fractional multiply) operations to start one microinstruction before a multiplication or multiply-accumulate (signed or unsigned) has been completed (e.g., one can start one multiply or multiply-accumulate once every three microinstructions). However, by doing that it is not possible to read the result in MACH and MACL (although the MDU flags can be tested), so this is intended to be used in a multiply-accumulate sequence. It is also allowed to mix different sizes in multiply/mac sequences.

Multiply-accumulate operations are similar to multiply operations, except that the contents of MACH and MACL registers are added to the multiplication result.

When multiply or multiply accumulate operations finish, MACL and MACH hold the least and the most significant 24-bit words, respectively.

37.15.8.3.2 Divide Operation Length

The divide operation is always unsigned. The division completes in 13 microcycles, meaning that after the start divide microinstruction, one has to wait for 12 microcycles and then read the result and the remainder in MACH and MACL registers. During the 12 execution microcycles, microengine can execute microinstructions unrelated to the MDU.

37.15.8.3.3 Signed Multiplication (mults)

MDU signed multiplication is defined as follows:

```
(signed) MACH,MACL = (signed) AS * (signed) BS
```

MC and MV flags are reset. MZ is set if result is 0, resets otherwise. MN is set if result is negative.

37.15.8.3.4 Unsigned Multiplication (multu)

MDU unsigned multiplication is defined as follows:

```
(unsigned) MACH|MACL = (unsigned) AS * (unsigned) BS
```

MC and MV flags are reset. MZ is set if result is 0, resets otherwise. MN is a copy of the most significant bit of result.

37.15.8.3.5 Signed Multiply-Accumulate (macs)

MDU signed multiply-accumulate is defined as follows:

```
(signed/unsigned) {MACH,MACL} += (signed) AS * (signed) BS
```

MC is not altered.

MV is set if result can not be represented by a 48-bit signed number. MACS never resets MV flag: it is left as is if no overflow occurs, or set it otherwise. This allows checking the overflow flag only once at the end of a series of multiply-accumulate operations in a scalar product calculation.

```
if (({MACH,MACL} += AS * BS < -247) || ({MACH,MACL} += AS * BS > 247 - 1))
    MV = 1
```

MZ is set if result is 0, resets otherwise. MN is a copy of the most significant bit of result.

Note that only 24-bit multiply-accumulate is available.

37.15.8.3.6 Unsigned Multiply-Accumulate (macu)

MDU Unsigned Multiply-Accumulate is defined as follows:

```
(signed/unsigned) {MACH,MACL} += (unsigned) AS * (unsigned) BS
```

MC is set if result can not be represented by a 48-bit unsigned non-negative number. MACU never resets MC flag: MC flag is left as is if no carry occurs, or set otherwise. This allows checking the carry flag only once at the end of a series of multiply-accumulate operations in a scalar product calculation.

```
if (({MACH,MACL} += AS * BS < 0) || ({MACH,MACL} += AS * BS > 248 - 1))
    MC = 1
```

MV is not altered.

MZ is set if result is 0, resets otherwise. MN is a copy of the most significant bit of result.

Note that only 24-bit multiply-accumulate is available.

37.15.8.3.7 Signed Fractional Multiplication (fmults)

MDU Signed Fractional Multiplication takes the B-Source as an **unsigned** 8- or 16-bit³⁵ fraction between 0 and $(2^8 - 1)/2^8$ (inclusive) for the 8-bit operation, or between 0 and $(2^{16}-1)/2^{16}$ (inclusive) for the 16-bit operation. Only A-Source is taken as a signed number. The value of B-Source is considered the unsigned numerator of a fraction with denominator 2^8 or 2^{16} for the 8- and 16-bit operations, respectively.

The integer part of the result is stored in MACH, and the fractional part in MACL. The result is signed, so that the concatenation of MACH and MACL form a 48-bit fixed point number with a 24-bit integer part, both for 8- and 16-bit operations. To calculate the unsigned numerator of the fractional part (with denominator 2^{24}) of the result, one must take the absolute value of MACL considering the signal of the result (not MACL alone), i.e.: if flag MN=1, invert MACL and add 1.

MDU flags are updated in the same way as in the Signed Multiplication.

37.15.8.3.8 Unsigned Fractional Multiplication (fmultu)

MDU Unsigned Fractional Multiplication takes both A-Source and B-Source as unsigned operands. B-Source is taken as an 8- or 16-bit³⁶ fraction between 0 and $(2^8 - 1)/2^8$ (inclusive) for the 8-bit operation, or between 0 and $(2^{16}-1)/2^{16}$ (inclusive) for the 16-bit operation. The value of B-Source is considered the numerator of a fraction with denominator 2^8 or 2^{16} for the 8- and 16-bit operations, respectively.

35. 24-bit fractional signed/unsigned multiplication is equivalent to mults/multu.

36. 24-bit fractional signed/unsigned multiplication is equivalent to mults/multu.

The integer part of the result is stored in MACH, and the fractional part in MACL. The fractional part in MACL is the numerator of a fraction with denominator 2^{24} . The concatenation of MACH and MACL form a 48-bit fixed point number with a 24-bit integer part, both for 8- and 16-bit operations.

MDU flags are updated in the same way as in the Unsigned Multiplication.

37.15.8.3.9 Unsigned Divide (div)

At the end of a divide operation MACL holds the result of the division, taking A-source as numerator and B-source as denominator, while MACH holds the remainder. If a divide by 0 is executed, MACL holds the maximum unsigned number (0xFFFFFFFF) as result and flag MV is set to indicate division by 0 (otherwise reset). The contents of MACH become indetermined.

MC flag is always reset.

MZ flag is set if MACL equals 0, and reset otherwise.

MN receives a copy of MACH bit 23 (msb from the remainder).

Note that signed division is not available.

37.15.8.3.10 MDU Flags

MDU has its own flags to indicate the result and status of an MDU operation. They are: MC, MZ, MV, MN and MB. All MDU flags are updated with the final result at the end of the operation, and do not change until the next operation finishes. Therefore it is possible to start a new MDU operation and test the flags of the previous one in parallel, except for mult/mac with 8-bit operand (takes only 1 microcycle).

37.15.8.3.10.1 MDU Negative Flag - MN

MN flag is always a copy of MACH bit 23 at the end of the operation, either in signed or unsigned ones. Note that MACH holds the rest of a division operation, which is always unsigned.

37.15.8.3.10.2 MDU Carry Flag - MC

MDU carry flag indicates if the result cannot be represented by a 48-bit number, in Signed and Unsigned Multiply Accumulates. It is reset in the other operations.

37.15.8.3.10.3 MDU Zero Flag - MZ

In multiply and multiply-accumulate operations, MDU zero flag is asserted if MACH and MACL are equal to zero at the end of an operation. In divide operations, zero flag is asserted if MACL (result) is equal to 0.

37.15.8.3.10.4 MDU Overflow Flag - MV

In multiply operations, MV flag is negated and keeps negated in the end, because the result of a multiplication can always fit in a 48-bit result (MACH and MACL concatenated). In a multiply-accumulate operation, MV is asserted if the result size is wider than 48 bits. MV flag work in both signed and unsigned operations.

In divide operations it is only asserted if a divide-by-zero operation was executed.

37.15.8.3.10.5 MDU Busy Flag (MB)

MB tests as true at the next microinstruction after the MDU start operation, and as false at the last microcycle of any MDU operation execution.

37.15.8.4 Branch conditions

Microengine allows conditional branch. There are five sets of flags that can be tested in a conditional branch: ALU flags, MDU flags, P flags, Channel flags, and Semaphore flag (flag SMLCK).

When a thread starts to be executed, the values in MDU and ALU flags are not initialized. ALU flags are described in [ALU flags](#), MDU flags are described in [MDU Flags](#). MDU and ALU flags are updated during execution of microinstructions.

P flags are actually the upper byte of P register, which optionally can work as user defined flags (see [P Register](#)).

Channel flags Flags0, Flag1, MRLA, MRLB, TDLA, TDLB, PSS, PRSS and PSTO are obtained from the selected channel (value in CHAN register), while channel flags, LSR, FM[0] and FM[1] are selected by the serviced channel, regardless of the CHAN value³⁷.

Flags TDLA/B, MRLA/B, LSR, FM and PSS, are sampled at the beginning of a thread. Flag PSS does not change during its execution while CHAN register is not written. When a write in CHAN register is performed, all flags except LSR and FM are updated according to the channel specified by CHAN value. Flags MRLA/B and TDLA/B are reset when their respective latches in channel are cleared by microcode.

37. Serviced channel does not change during execution of a thread, and it is the channel that requested a service (initial value of CHAN register when a thread starts).

Table 37-369. Channel flags as branch condition

Flag	Description	Service or selected channel
Flag0	State Resolution flag	reflects the selected channel (CHAN)
Flag1	State Resolution flag	reflects the selected channel (CHAN)
MRLA	Match A Recognition Latch	these flags reflect the selected channel (CHAN)see MRLA/B - Match Recognition Latches and TDLA/B - Transition Detect Latches for more information.
MRLB	Match B Recognition Latch	
TDLA	Transition A Detection Latch	
TDLB	Transition B Detection Latch	
LSR	Link Service Request	reflects the serviced channel.
PRSS	Input Pin state sampled at service request	reflects the selected channel (CHAN). Does not change if CHAN is not changed.
PSS	Sampled Input Pin State	reflects the selected channel (CHAN). Does not change if CHAN is not changed (see Pin Control Registers).
PSTI	Current Input Pin State.	reflects the selected channel (CHAN). Changes any time.
PSTO	Current Output Pin State	reflects the selected channel (CHAN). Changes any time.
FM	Function Mode Bits	reflects the Function Mode for serviced channel (see eTPU Channel n Status Control Register for Engine 1 (eTPU_CCCR_CnSCR_ENG1) and eTPU Channel n Status Control Register for Engine 2 (eTPU_CCCR_CnSCR_ENG2))

Semaphore condition SMLCK always indicates if a semaphore is locked for the Engine, resolving as false before any lock attempt. For each trial, the SMLCK flag is updated. The SMLCK value set in one thread is not meaningful to the other. After a free, the SMLCK condition tests as false until a new lock attempt on the same thread.

Branch conditions are selected through instruction fields BCC and BCF (see [Conditional/unconditional branch](#)).

37.15.9 Microinstruction Set

Each microinstruction can execute up to 3 microoperations in parallel. Microinstructions are grouped into formats, and there are four types of microoperations:

- ALU/MDU Operations
- SDM Operations
- Channel Configuration/Control Operations
- Flow Control Operations

Each microinstruction format is defined by a set of microinstruction fields, which determine the operations, each belonging to one of the groups above (there may be several in one group). Complete microinstruction formats are shown in [Microinstruction formats](#).

Parallelism conflicts may arise when two operations are executed in the same microinstruction. These situations are explained in [Microinstruction Parallelism Issues](#).

37.15.9.1 SDM Microoperations

The access to SDM is made by providing an address and a register to perform a data transfer, except semaphore operations, which are also classified in the SDM group. Only P and DIOB registers can exchange data with SDM. Microengine always addresses SDM in 32-bit boundaries, for 8, 24, or 32 bit wide data.

Direction is determined by the field RW in all addressing modes: RW=0 selects read and RW=1 selects write.

37.15.9.1.1 SDM Addressing Modes

There are 4 addressing modes in eTPU:

- Absolute
- Selected Channel Relative
- Indirect
- Engine Relative

The addressing modes Absolute and Selected Channel Relative use immediate bits to form the physical address of SDM, which is identified in microinstruction as a field called AID. AID field can be 3-, 7-, or 8-bit wide depending on the addressing mode.

37.15.9.1.1.1 Absolute Addressing Mode

In Absolute addressing mode, the address range is 256 parameters, addressed by field AID, which in this mode is 8-bit wide. These parameters are located in SDM addresses from 0 to 255.

```
physical_address = AID[7:0]
```

37.15.9.1.1.2 Selected Channel Relative Addressing Mode

In Selected Channel Relative addressing mode, only the first 8 (with 3-bit AID) or 128 (with 7-bit AID) parameters of the selected channel are accessible, depending on the microinstruction format. Physical address is calculated using the channel parameter base address that is specified in the CPBA field of [eTPU Channel n Configuration Register for Engine 1 \(eTPU_CCCR_CnCR_ENG1\)](#) and [eTPU Channel n Configuration Register for Engine 2 \(eTPU_CCCR_CnCR_ENG2\)](#). AID field is added to channel parameter base address to compose the physical address. The address equation is:

```
physical_address = selected_channel_parameter_base_address + AID[6:0], or
physical_address = selected_channel_parameter_base_address + AID[2:0]
```

37.15.9.1.1.3 Indirect Addressing Mode

In Indirect Addressing mode the physical address is taken from DIOB register. Only DIOB bits 14 to 2 are relevant. Since the SDM word address is shifted two bits up in DIOB, its contents hold the same parameter address value used by Host. The address equation for SDM access is:

```
physical_address = DIOB[13:2], or
physical_address = (truncated to 12 bits) (DIOB / 4)
```

Indirect addressing mode is used to access the Shared Subsystem Area when `eTPU_CCCR_CxCR_ENGn[SSMAE]=1` and DIOB addresses the 16 Kbytes just above the SDM range (DIOB bit 14 = 1). See [Shared Subsystems support](#) for more details. If `SSMAE=0`, accesses with DIOB bit 14 = 1 hit the same SDM addresses as with bit 14 = 0, providing full backwards compatibility with respect to DIOB contents.

Indirect addressing mode can have post-increment or pre-decrement on DIOB, allowing stack operations. See [DIOB stack operation](#) for more information.

37.15.9.1.1.4 Engine Relative Addressing Mode

In Engine Relative Addressing mode the physical address is the concatenation of the `eTPU_SCR_ECR_ENGn[ERBA]` field (see [eTPU Engine 1 Configuration Register \(eTPU_SCR_ECR_ENG1\)](#) and [eTPU Engine 2 Configuration Register \(eTPU_SCR_ECR_ENG2\)](#)) with the 7-bit AID instruction field. This allows the same function microcode, when running on distinct engines, to access different address spaces, global to the engine only.

The SDM physical word address in Engine Relative Addressing mode is calculated as follows:

```
physical address = (eTPU_SCR_ECR_ENGn[ERBA] << 7) + AID[6:0]
```

37.15.9.1.2 SDM source/destination registers

When performing an SDM operation, only DIOB or P can be used as data source or destination. P is 32-bit wide, and DIOB is 24-bit wide. Microinstruction field P/D (1 bit) is used to choose between P and DIOB as data the source or destination. When the P/D field is not available in microinstructions that support SDM access, the source/destination is P.

Table 37-370. SDM source/destination Register Selection

P/D	Meaning
0	P access
1	DIOB access

37.15.9.1.3 SDM Operation Size

When using DIOB register to perform SDM data transfers, the operation size is always 24-bit wide (lower 24 bits of SDM). When using P register, the operation size can be 8-, 24- or 32-bit wide, which is controlled by microcode RSIZ field (2 bits). RSIZ meaning is shown in the following table.

Table 37-371. SDM P access Size

RSIZ	P access	DIOB access
00	full 32-bit access (i.e. P[31:0]=SDM[addr] [31:0])	RESERVED
01	only upper 8 bits are transferred (i.e. P[31:24] = SDM[addr] [31:24])	RESERVED
10	only lower 24 bits are transferred (i.e. P[23:0] = SDM[addr] [23:0])	DIOB = SDM[addr] [23:0]
11	RESERVED	RESERVED

RSIZ is not available in all microinstructions that support SDM access. In microinstructions where RSIZ field is not available, SDM access will be 24 bits by default.

When performing a Zero SDM write operation (see [Zero SDM Operation](#)), RSIZ defines the size of operation regardless of the P/D field ([SDM source/destination registers](#)).

37.15.9.1.4 SDM access direction

RW field defines the direction of the access in the SDM. The access direction is summarized in the following table.

Table 37-372. SDM access direction

R/W	Meaning
0	read SDM parameter into P or DIOB registers
1	write SDM parameter from P or DIOB registers

37.15.9.1.5 Zero SDM Operation

Zero SDM operation is controlled by microcode field ZRO (1bit). When ZRO field is 0, the data portion written in SDM or in P/DIOB (SDM read) registers will always be 0x0. When performing a Zero SDM write operation, the RSIZ is relevant regardless of the P/D field (usually RSIZ is meaningful only for P/D = 0), which means that zero SDM write operation can be performed with 32, 24 or 8 bits according to SDM operation size. These conditions are summarized in the following table.

Table 37-373. Zero SDM Operation

ZRO	RW	P/D	Meaning
0	0	0	Clear P register. Size is determined by RSIZ field. See SDM Operation Size
0	0	1	Clear DIOB (all 24 bits), independently of RSIZ
0	1	x	Clear SDM parameter. Size is determined by RSIZ field. See SDM Operation Size
1	RW	P/D	Regular SDM operation

Note

When field STC is present, STC=11 also disables Zero SDM operation (see [Table 37-374](#)). The conflicts with DIOB operations (see [DIOB stack operation](#)) and ALU operations are resolved like a normal SDM operation (see [Table 37-415](#)).

Zero SDM read operations do check memory errors on the read data, however discarded. Zero SDM write operations normally update the ECC of the written data.

37.15.9.1.6 DIOB stack operation

SDM Indirect Addressing Mode (see [Indirect Addressing Mode](#)) is used if STC field (2 bits) exists in the microinstruction, controlling automatic increment/decrement of DIOB register, as shown in the following table, thus allowing stack operations. DIOB is incremented and decremented in word addresses, only from bits 15 downto 2, i.e.: the bits 23 to 16 and 1 to 0 are left untouched by STC pre-decrement and post-increment.

Table 37-374. DIOB Post-Increment / Pre-Decrement - STC

STC	Meaning
00	Post-Increment of DIOB
01	Pre-Decrement of DIOB
10	No Increment/Decrement (normal access)
11	No SDM Access ¹

1. Also disables Zero SDM operation.

37.15.9.1.7 Semaphore Operations

Semaphore lock and free operations are available through eTPU microcode. For more information about semaphores see [Hardware Semaphores](#). There are 2 microinstruction fields that control semaphore operations: FL (1 bit) and SMPR (2 bits). Serviced channel sees 4 semaphores, selected by field SMPR.

Table 37-375. Semaphore Operations Fields

Field	Meaning
FL	0 = free semaphore, 1 = lock semaphore
SMPR	semaphore number selector

When freeing a semaphore, the field SMPR has no meaning. This is because only one semaphore can be locked at a time by each engine, so when freeing a semaphore it is not necessary to specify its number.

Note

If microcode tries to lock a semaphore already locked for the same Engine, the semaphore continues locked for the Engine and the SMLCK branch condition resolves as true.

37.15.9.2 ALU/MDU Operations

ALU/MDU microoperations mostly comprises two sources, one destination and one operation. The operation is generally selected through fields ALUOP, ALUOPI or SHF. In formats where there is no operation selection field (ALUOP, ALUOPI or SHF), the operation performed is always addition; however, it is possible to perform subtraction, increment or decrement using fields BINV (see [B-Source Inversion](#)) and CIN (see [Carry-in Control](#)).

37.15.9.2.1 Source and Destination Register Set Selection

Microcode field T4ABS allows selection of a source from either one of two register sets, shown in [Table 37-379](#). The same applies to T2ABD, used for ALU destination selection with other two register sets, as shown in [Table 37-380](#). Microinstruction fields ABSE and ABDE control the register set selection for source and destination, respectively, when available at the format. In formats without ABSE/ABDE, the field T4BBS determines the register sets used by T2ABD and T4ABS, as shown in [Microinstructions without fields ABSE and ABDE](#).

37.15.9.2.1.1 Microinstructions With Fields ABSE and ABDE

In microinstructions where ABSE and ABDE fields are available (1 bit each), ABSE controls register set selection for T4ABS and ABDE controls register set selection for T2ABD. The following table shows the meaning of values for ABSE and ABDE fields.

Table 37-376. Register Set Selection by ABSE or ABDE

ABSE or ABDE	Register Set Selected
0	second
1	first

37.15.9.2.1.2 Microinstructions without fields ABSE and ABDE

When ABSE and ABDE are not available in a microinstruction format, register set is selected by T4BBS field. The following table explains how to select a register set either for T4ABS and T2ABD.

Table 37-377. Register set selection by T4BBS without ABSE and ABDE

T4BBS	Register set for T2ABD	Register set for T4ABS
0xx	first	first
100	second	second
101	second	first
110	first	second
111	first	first
none ¹	first	first

1. Refers to operations with immediate data as B-source, without ABSE and ABDE.

37.15.9.2.2 Selecting sources and destination

All ALU/MDU operations need 2 sources (called AS and BS) and 1 destination (called AD), except for some of those that use immediate data (see [Operations With Immediate Data](#)). Fields T4ABS (4 bits), ABSE (1 bit), T4BBS (3 bits) select sources, while T2ABD (4 bits) and ABDE (1 bit) select the destination. When MDU is used (multiply/divide), T2ABD destination selection is ignored and results are stored in MACH and MACL (see [MAC and Divide Unit \(MDU\)](#)). ABSE and ABDE are not available in some microinstruction formats that support ALU/MDU operations. However, in all formats where ABSE is available, ABDE is also available and vice-versa. The existence of ABSE/ABDE fields changes the meaning of T4BBS field, as shown in the table below. On instructions with immediate data, it is used as B-source (see [Operations With Immediate Data](#)).

All sources and destinations have a size associated to it, and these sizes are used to select flag sample position (see [Flags Sampling Control](#)). Sizes can be 8, 16 or 24 bits. Registers that are not exactly of one of these sizes are treated as the immediately upper size (e.g., CHAN[4:0] is an 8-bit source). See [Flags Sampling Control](#) for more information.

Some parallelism issues arise when selecting P, DIOB, ERTA or ERTB as destination registers, since they can be modified by other microoperations in the same microinstruction (see [Microinstruction Parallelism Issues](#) for details).

Table 37-378. B source selection - T4BBS

T4BBS	Meaning in microinstruction formats with ABSE/ABDE	Meaning in microinstruction formats without ABSE/ABDE ¹
000	BS[23:0] = P[23:0]	
001	BS[23:0] = A[23:0]	
010	BS[23:0] = SR[23:0]	
011	BS[23:0] = DIOB[23:0]	
100	reserved	BS = 0
101	reserved	BS = 0
110	reserved	BS = 0
111	BS=0, or Max const., if CIN=0 and BINV=0 (see Generating "max" constant).	

1. T4BBS also selects A-source and destination register set in this case, according to [Table 37-377](#).

T4ABS selects one source from 2 register sets, shown in the following table. ABSE and T4BBS control which set T4ABS field uses to select the source. For more information about how to select a register set for T4ABS and T2ABD see [Source and Destination Register Set Selection](#). All sources are zero-filled to 24 bits, unless sign-extension is specified (see [A-Source Size Override](#)).

Table 37-379. A source selection - T4ABS

T4ABS	First register set		Second register set	
	Selected register	Size	Selected Register	Size
0000	AS[7:0]=P[7:0]	8	AS[7:0]=0	8
0001	AS[7:0]=P[15:8]	8	AS[23:0]=C[23:0]	24
0010	AS[7:0]=P[31:24]	8	AS[15:0] = TPR[15:0]	16
0011	AS[23:0] = ERTB[23:0]	24	AS[23:0] = B[23:0]	24
0100	AS[23:0] = D[23:0]	24	AS[23:0] = TRR[23:0]	24
0101	AS[15:0] = P[15:0]	16	AS[7:0] = 0, read_match ¹	8
0110	AS[15:0] = P[31:16]	16	AS[13:0] = RAR[13:0]	16
0111	AS[7:0] = P[23:16]	8	AS[23:0] = MACH[23:0]	24
1000	AS[23:0] = P[23:0]	24	AS[23:0] = MACL[23:0]	24
1001	AS[23:0] = A[23:0]	24	AS[4:0]=CHAN[4:0]	8
1010	AS[23:0] = SR[23:0]	24	AS[14 :2] = CHAN_BASE ²	16
1011	AS[23:0] = DIOB[23:0]	24	AS[13:0] = ENGINE_BASE ³	16
1100	AS[23:0] = TCR1[23:0]	24	Reserved	-
1101	AS[23:0] = TCR2[23:0]	24	Reserved	-
1110	AS[23:0] = ERTA[23:0]	24	Reserved	-
1111	AS[23:0] = 0	24	Reserved	-

1. T4ABS=0101 with second register set also reads MatchA/B registers into ERTA/B (see [Special T4ABS Source Operation: Read Match Registers](#)).
2. CHAN_BASE is the selected channel's base SDM address in channel relative address mode (see [CHAN_BASE as a source](#)).
3. ENGINE_BASE is the eTPU_SCR_ECR_ENGn[ERBA] field shifted left 9 positions.

T2ABD selects the destination from 1 of 2 register sets, shown in the following table. ABDE and T4BBS control which set T2ABD field uses to select the destination.

Table 37-380. Destination selection - T2ABD

T2ABD	First register set		Second register set	
	Selected register	Size	Selected Register	Size
0000	A[23:0] = AD[23:0]	24	C[23:0] = AD[23:0]	24
0001	SR[23:0] = AD[23:0]	24	LINK[7:0] = AD[7:0]	8
0010	ERTA[23:0] = AD[23:0] ¹	24	TPR[15:0] = AD[15:0]	16
0011	ERTB[23:0] = AD[23:0] ²	24	B[23:0] = AD[23:0]	24
0100	DIOB[23:0] = AD[23:0]	24	CHAN[4:0] = AD[4:0]	8
0101	P[15:0] = AD[15:0]	16	D[23:0] = AD[23:0]	24
0110	P[31:16] = AD[15:0]	16	RAR[12:0] = AD[12:0]	16
0111	P[23:0] = AD[23:0]	24	MACH[23:0] = AD[23:0]	24
1000	TCR1[23:0] = AD[23:0]	24	MACL[23:0] = AD[23:0]	24
1001	TCR2[23:0] = AD[23:0]	24	Reserved	-
1010	P[31:24] = AD[7:0]	8	Reserved	-

Table continues on the next page...

Table 37-380. Destination selection - T2ABD (continued)

T2ABD	First register set		Second register set	
	Selected register	Size	Selected Register	Size
1011	P[23:16] = AD[7:0]	8	Reserved	-
1100	P[15:8] = AD[7:0]	8	Reserved	-
1101	P[7:0] = AD[7:0]	8	Reserved	-
1110	TRR[23:0] = AD[23:0]	24	Reserved	-
1111	no destination selected ³	24	Reserved	-

1. T2ABD=0010 with first register set also writes to MatchA or UDCM registers of the selected channel if field ERWA=0 (see [Write Channel Match and UDCM Registers](#)).
2. T2ABD=0011 with first register set also writes to MatchB register of the selected channel if field ERWB=0.
3. if no destination is selected, ALU flags are updated, although the result is lost.

37.15.9.2.2.1 Max Const Generation With T4BBS=111

When T4BBS = 111, BINV = 0, and CIN = 0, the value assigned to BS will be 0x800000, and not 0x0 as expected. See [Generating "max" constant](#) for detailed explanation about this.

37.15.9.2.2.2 Special T4ABS Source Operation: Read Match Registers

When T4ABS = 0101 and the source for T4ABS is selected from the second register set, the constant 0x00 is used as AS (8-bit size) and the following register transfer is performed in parallel as well: match registers of the selected channel (value in CHAN register) are copied to ERTA/ERTB registers, where ERTA receives the value of MatchA register and ERTB receives the value of MatchB register (see [ER - Event Registers](#)). Note that ALU destination can still be chosen by T2ABD in parallel. When ERTA or ERTB is selected by T2ABD, a parallelism issue arises (see [ALU Operations and Read Match Registers](#)).

37.15.9.2.2.3 CHAN_BASE as a source

Each channel has a parameter base address in SDM, which is configured in the CPBA field of [eTPU Channel n Configuration Register for Engine 1](#) (eTPU_CCCR_CnCR_ENG1) and [eTPU Channel n Configuration Register for Engine 2](#) (eTPU_CCCR_CnCR_ENG2). CHAN_BASE, which represents a parameter address (CPBA*2), can be used as A-source using T4ABS=1010 when T4ABS selects a source from the second register set. In this case, CHAN_BASE is loaded into AS[13:2] to form the byte address (AS[23:14]=0, AS[1:0]=0). For example, in Indirect addressing mode, where the destination register is DIOB, CHAN_BASE is loaded into DIOB[13:2], which is the parameter address, and DIOB[13:0] represents the byte address. CHAN_BASE is the base address of the selected channel (given by CHAN register).

37.15.9.2.3 Flags Sampling Control

This section explains how the flags Z (zero), C (carry), N (negative) and V (overflow) are updated in an ALU operation. When there are post-ALU shift operations, the ALU Carry Out is not directly sampled in Carry flag, but passed to the post-ALU shifter (see [Shift Operations](#)). Since the size of source operands in ALU operations is variable, flags can be sampled as an operation of 8, 16 or 24 bits wide. The operation size selection is automatic, based on defined sizes of sources and destination, using the equation:

```
operation_size = minor(size_of(destination), greater(size_of(A-Source),
size_of(B-Source)))
```

Operation size can also be shown with the following table:

Table 37-381. Operation Size Determination

A Source	B Source	Destination	Operation Size
x	x	8 bits	8 bits
8 bits	8 bits	x	8 bits
16 or 24 bits	x	16 bits	16 bits
16 bits	8 or 16 bits	24 bits	16 bits
8 or 16 bits	16 bits	24 bits	16 bits
24 bits	x	24 bits	24 bits
x	24 bits	24 bits	24 bits

Note

Whenever BS = (constant) 0, its size is considered 8 bits, and all 24 bits in B-bus are set to 0. Therefore, all operations with BS = (constant) 0 have their size determined by AS and Destination only.

CCS field (1 bit) controls whether flags will be updated or not ([Table 37-383](#)). When CCS bit exists in a microinstruction, the operation size will be used to sample flags. In some microinstructions CCS field is replaced by CCSV (2 bits, [Table 37-382](#)). Flag sampling according to CCSV can be set as defined by the operation size, or fixed as 8 or 16 bit operations.

Note

For the post-ALU rotate right operation, CCSV also determines the rotate size: whether 8-bit, 16-bit, or determined by the operation size.

Table 37-382. Flag Sampling Using CCSV field

CCSV	Meaning
00	sample flags as an 8 bit operation
01	sample flags as a 16 bit operation
10	sample flags as defined by operation size
11	do not sample flags

Table 37-383. Flag Sampling Using CCS field

CCS	Meaning
0	sample flags as defined by operation size
1	do not sample flags

When neither CCS nor CCSV are present in the microinstruction, flags are not sampled. CCS and CCSV do not affect the Carry update on Exchange Bit operation (see [Exchange Bit](#)), but does control the N and Z flags.

37.15.9.2.4 B-Source Inversion

The data selected as second source (T4BBS) can be inverted (bitwise boolean NOT) before operation. This is controlled by microinstruction field BINV (1 bit, in the following table). A zero value for BINV activates B-source inversion.

Table 37-384. B-Source Inversion - BINV

BINV	Meaning
0	invert B-source ¹
1	keep B-source bus unchanged

1. Except on max-constant selection, see [Generating "max" constant](#).

BINV also selects between *adc* or *sbc* enhanced ALU operation, using inverted C flag as carry-in besides BS inversion for *sbc*. Note that BINV does not invert carry in fixed-carry operations (see [Table 37-385](#)).

When BINV = 0, T4BBS = 111 and CIN = 0, the value assigned to BS is 0x800000, instead of 0x0. See [Generating "max" constant](#) for more details.

37.15.9.2.5 Carry-in Control

CIN field (1 bit, [Table 37-385](#)) controls the carry-in for addition/subtraction operations. Functionality of CIN field depends on the arithmetic operation selected by ALUOP. When ALUOP is not available in microinstruction, the operation selected is *add*. For carry-in control in MDU operations, see the following table.

Table 37-385. ALU Carry-In Control

operation	CIN=0	CIN=1
<i>add</i> (addition)	carry-in used is 1 ¹	carry-in used is 0
<i>adc</i> (addition with carry) ²	carry-in used is C flag	
<i>sbc</i> (subtraction with borrow) ³	carry-in used is inverted C flag	

1. Except on max-constant selection, see [Generating "max" constant](#).

2. Selected by ALUOP=11000 and BINV=1

3. Selected by ALUOP=11000 and BINV=0

37.15.9.2.5.1 Generating "max" constant

When T4BBS = 111, CIN = 0 and BINV = 0, BS is assigned to 0x800000 (called "max constant") instead of 0x000000. This is an exception for CIN and BINV fields: when "max constant" is selected, the carry in is 0 and B-source ("max constant" itself) is not inverted, neither the carry out.

"Max constant" is the value which, added to a time base value minus 1, gives the farthest wrapped time base value that satisfies a channel greater-equal comparison. See [Enhanced Channels](#) for more info.

37.15.9.2.6 Shift Operations

There are three types of shift operations: ALU, post-ALU and Shift Register. ALU shift operations are covered in [ALU/MDU Operation Selection](#). Post-ALU and Shift Register are covered in the following sections.

37.15.9.2.6.1 Shift Register Operations

SR can be used as a general purpose register and it can easily shift-right its contents, combined or not to post-ALU shift operations. If field SRC (1 bit) in microcode is 0, SR will shift its contents 1 bit to the right according to the algorithmic description below. SR shifting operation depends also on SHF or ALUOP fields. ALUOP and SHF never exist both on the same microinstruction format.

Table 37-386. Shift Register Control - SRC

SRC	Meaning
0	shift right SR 1 bit
1	no shift

SR Operation:

```

SR[22:0] = SR[23:1];
if SHF == "01" or ALUOP == "10110" then
    SR[23] = ALU_OUT[0];
else
    SR[23] = 0;
endif;

```

37.15.9.2.6.2 Post-ALU shift operations

Post-ALU shift can be selected by SHF field (2 bits) or by some specific ALUOP field values. SHF and ALUOP fields are never both available in the same microinstruction format. When selecting post-ALU shift operation using ALUOP field, ALU will always add the sources before shifting the result.

Table 37-387. Post-ALU shift operation

Post ALU operation	SHF ¹	ALUOP
shift left (1 bit)	00	10101
shift right (1 bit)	01	10110
rotate right (1 bit)	10	10111
no shift/rotate	11	any other ²

1. ALU performs AS+BS before shift/rotate for all SHF values.
2. Some ALUOP combinations perform shift/rotate, but not using the Post-ALU Shifter (see [Table 37-392](#))

Carry flag is only updated when CCS or CCSV fields allow it (see [Flags Sampling Control](#)). Algorithmic descriptions of post-ALU shift operations are presented in [ALU ADD operation with and without shifting](#).

37.15.9.2.7 Conditional ALU/MDU Operation Execution

The 3-bit field AS/CE allows conditional execution of arithmetic operation, as shown in the following table. The same field can also be used for overriding the size of A-Source (see [A-Source Size Override](#)).

Table 37-388. ALU/MDU Conditional Execution

AS/CE	Meaning
000	used for A-Source size override (see A-Source Size Override)
001	
010	execute if C = 1
011	execute if C = 0
100	execute if Z = 1
101	execute if Z = 0
110	execute if N = 1
111	execute unconditionally/no size override

Other operations not related to ALU/MDU in the same microinstruction are not affected by the AS/CE field.

If a conditional operation is selected, there is no A-Source size override; similarly, when size override for A-Source is selected, the ALU/MDU operation executes unconditionally.

When a conditional ALU/MDU operation is not executed:

- the destination register is not updated. If the destination is CHAN, no actions associated with CHAN assignment occur (see [Channel Selection Register - CHAN](#)).
- the ALU and MDU flags are not updated.
- MDU does not start any operation, i.e., MACH and MACL are not updated.
- SR does not shift.
- T4ABS-selected read-match does not occur.

37.15.9.2.8 A-Source Size Override

Some values if the AS/CE field are used for A-Source Size Override, as shown in the following table.

Table 37-389. A-Source Size Override

AS/CE	Meaning
000	A-source size override to 8 bits
001	A-source size override to 16 bits

Table continues on the next page...

Table 37-389. A-Source Size Override (continued)

AS/CE	Meaning
010	used for conditional execution (see Conditional ALU/MDU Operation Execution)
011	
100	
101	
110	
111	execute unconditionally/no size override

Register size override zero-pads an overridden source to 24-bits (if no sign extension is performed, see [A-source Sign Extension](#)) and affects operation size calculation. When register source is wider than size override, most significant bits of selected register are not used as A source (zeros are used instead). When size override is wider than selected register, register value is padded to zeros.

When size override is used with MDU operations, it affects only the operand values, but not the operation size: MDU operation size is fully determined by the operation definition (fields ALUOP, ALUOPI).

Table 37-390. AS/CE field A source size override functionality

Size Override	Size of Selected Register	AS Value ¹
8 bits	8 bits	reg[7:0]
8-bits	16 bits	reg[7:0]
8bits	24 bits	reg[7:0]
16 bits	8 bits	reg[15:0]
16 bits	16 bits	reg[15:0]
16 bits	24 bits	reg[15:0]

1. All values are zero-padded to 24 bits

37.15.9.2.9 A-source Sign Extension

The SEXT microinstruction field forces sign extension of A source according to the size of A operand, either overridden or not by AS/CE field, according to the following table. The sign is taken from the size-overridden value, not the original one.

Table 37-391. A source Sign Extension

SEXT	Meaning
0	extends sign of A source from its size to 24-bits
1	does not extend sign of A source

A-source sign is not extended in microinstructions without SEXT field, even if AS/CE field is present.

37.15.9.2.10 ALU/MDU Operation Selection

When field ALUOP is available in microinstruction, enhanced ALU operations shown in the following table can be performed, otherwise addition is performed. The ALU operations are defined in [ALU and Post-ALU Shifter](#). The MDU operations are defined in [MAC and Divide Unit \(MDU\)](#).

Table 37-392. ALU Operation Selection - ALUOP

ALUOP	Operation	Comment
00000	AS mults BS[7:0]	signed multiplication
00001	AS multu BS[7:0]	unsigned multiplication
00010	AS fmults BS[7:0]	signed fractional multiplication
00011	AS fmultu BS[7:0]	unsigned fractional multiplication
00100	AS mults BS[15:0]	signed multiplication
00101	AS multu BS[15:0]	unsigned multiplication
00110	AS fmults BS[15:0]	signed fractional multiplication
00111	AS fmultu BS[15:0]	unsigned fractional multiplication
01000	AS mults BS[23:0]	signed multiplication
01001	AS multu BS[23:0]	unsigned multiplication
01010	AS macs BS[23:0]	signed multiply-accumulate
01011	AS macu BS[23:0]	unsigned multiply-accumulate
01100	AS div BS[7:0]	unsigned division by 8-bit value
01101	AS div BS[15:0]	unsigned division by 16-bit value
01110	AS div BS [23:0]	unsigned division by 24-bit value
01111	n.a.	RESERVED
10000	AS[23:0] BS[23:0]	24 bit bitwise OR
10001	AS[23:0] ^ BS[23:0]	24 bit bitwise XOR
10010	AS[23:0] & BS[23:0]	24 bit bitwise AND
10011	abs(AS)	absolute value of AS
10100	AS + BS	arithmetic addition
10101	(AS + BS) shl 1	arithmetic addition with 1-bit post-ALU shift left. (Post-ALU shift operations)
10110	(AS + BS) shr 1	arithmetic addition with 1-bit post-ALU shift right (Post-ALU shift operations)
10111	(AS + BS) ror 1	arithmetic addition with 1-bit post-ALU rotate right (Post-ALU shift operations)
11000	AS adc/sbc BS ¹	addition/subtraction with C flag (Carry-in Control)
11001	AS shl (2 ^{^(BS[1:0]+1)})	AS is shifted left: 2 bits for BS=0; 4 for BS=1; 8 for BS=2; 16 for BS=3
11010	AS shr (2 ^{^(BS[1:0]+1)})	AS is shifted right: 2 bits for BS=0; 4 for BS=1; 8 for BS=2; 16 for BS=3
11011	AS ror (2 ^{^(BS[1:0]+1)})	AS is rotated right: 2 bits for BS=0; 4 for BS=1; 8 for BS=2; 16 for BS=3
11100	AS exch BS[4:0]	exchange C flag and AS bit determined by BS (Exchange Bit)
11101	AS setb BS[4:0]	set bit in AS determined by BS ²

Table continues on the next page...

Table 37-392. ALU Operation Selection - ALUOP (continued)

ALUOP	Operation	Comment
11110	AS clrb BS[4:0]	clear bit in AS determined by BS ³
11111	n.a.	RESERVED

1. Addition/Subtraction is selected by field BINV (see [B-Source Inversion](#))
2. In setb and clrb operations, the register that drives A source is not changed, unless selected as destination of the operation.
3. In setb and clrb operations, the register that drives A source is not changed, unless selected as destination of the operation.

37.15.9.2.11 Operations With Immediate Data

Immediate data can be used with some specific microinstruction formats. eTPU microcode allows 8-, 16- or 24-bit immediate data. Immediate data is loaded as B-source, so T4BBS field is not available. When using 24-bit immediate, an add is performed with A-source = 0 and flags are not updated. ALU operations are available with 8-bit immediate, although the field that selects ALU operation in this case is ALUOPI.

37.15.9.2.11.1 24-bit Immediate Destination

When using 24-bit immediate data, the destination register is selected by T2D field (2 bits), according to the following table.

Table 37-393. 24-bit Immediate Destination - T2D

T2D	Target Register
00	P[23:0]
01	A
10	SR
11	DIOB

37.15.9.2.11.2 Enhanced ALU Operations With Immediate Data

Enhanced operations with immediate data, selected by ALUOPI (5 bits) are allowed only with an 8 bit immediate operand (see the following table).

Table 37-394. ALU Operation Selection With Immediate Data - ALUOPI

ALUOPI	Operation	Comment
00000	AS mults #imm8	signed multiplication
00001	AS multu #imm8	unsigned multiplication
00010	AS fmults #imm8	signed fractional multiplication
00011	AS fmultu #imm8	unsigned fractional multiplication

Table continues on the next page...

Table 37-394. ALU Operation Selection With Immediate Data - ALUOPI (continued)

ALUOPI	Operation	Comment
00100	AS div #imm8	unsigned division
00101	n.a.	reserved
00110	n.a.	reserved
00111	n.a.	reserved
01000	AD[7:0] = AS[7:0] #imm8, AD[23:8] = AS[23:8]	bitwise OR
01001	AD[7:0] = AS[7:0] ^ #imm8, AD[23:8] = AS[23:8]	bitwise XOR
01010	AD[7:0] = AS[7:0] & #imm8, AD[23:8] = AS[23:8]	bitwise AND
01011	AD[7:0] = AS[7:0] & #imm8, AD[23:8] = 0x0	bitwise AND with clear
01100	AD[15:8] = AS[15:8] #imm8, AD[23:16] = AS[23:16], AD[7:0] = AS[7:0]	bitwise OR
01101	AD[15:8] = AS[15:8] ^ #imm8, AD[23:16] = AS[23:16], AD[7:0] = AS[7:0]	bitwise XOR
01110	AD[15:8] = AS[15:8] & #imm8, AD[23:16] = AS[23:16], AD[7:0] = AS[7:0]	bitwise AND
01111	AD[15:8] = AS[15:8] & #imm8, AD[23:16] = 0x0, AD[7:0] = 0x0	bitwise AND with clear
10000	AD[23:16] = AS[23:16] #imm8, AD[15:0] = AS[15:0]	bitwise OR
10001	AD[23:16] = AS[23:16] ^ #imm8, AD[15:0] = AS[15:0]	bitwise XOR
10010	AD[23:16] = AS[23:16] & #imm8, AD[15:0] = AS[15:0]	bitwise AND
10011	AD[23:16] = AS[23:16] & #imm8, AD[15:0] = 0x0	bitwise AND with clear
10100	AS + #imm8	arithmetic addition
10101	(AS + #imm8) shl 1	arithmetic addition with 1-bit shift left.
10110	(AS + #imm8) shr 1	arithmetic addition with 1-bit shift right
10111	(AS + #imm8) ror 1	arithmetic addition with 1-bit rotate right
11000	n.a.	reserved
11001	AS shl (2^(#imm8[1:0]+1))	AS is shifted left: 2 bits for #imm8=0; 4 for #imm8=1; 8 for #imm8=2; 16 for #imm8=3
11010	AS shr (2^(#imm8[1:0]+1))	AS is shifted right: 2 bits for #imm8=0; 4 for #imm8=1; 8 for #imm8=2; 16 for #imm8=3

Table continues on the next page...

Table 37-394. ALU Operation Selection With Immediate Data - ALUOPI (continued)

ALUOPI	Operation	Comment
11011	AS ror ($2^{(\#imm8[1:0]+1)}$)	AS is rotated right: 2 bits for #imm8=0; 4 for #imm8=1; 8 for #imm8=2; 16 for #imm8=3
11100	AS exch #imm8[4:0]	exchange C flag and AS bit determined by #imm8[4:0] (see Exchange Bit)
11101	n.a.	reserved
11110	n.a.	reserved
11111	n.a.	reserved

37.15.9.3 Channel Control and Configuration Microoperations

Channel Control and Configuration fields set configuration values in the channel logic of the channel selected by the CHAN register, except fields LSR and CIRC.

37.15.9.3.1 Channel flags operations

Each channel has two associated hardware flags, called Channel Flag 0 and Channel Flag 1. Microcode field FLC (3 bits) allows them to be set or cleared, as shown in the following table. These flags can be tested by microcode as a branch condition, and may also influence in the Entry Point taken, allowing fast state decoding. For more details, see [Entry Points](#).

Table 37-395. P flags operation - FLC

FLC	Meaning
000	Clear flag0
001	Set flag0
010	Clear flag1
011	Set flag1
100	Copy flag1:flag0 from P[25:24]
101	Copy flag1:flag0 from P[27:26]
110	Copy flag1:flag0 from P[29:28]
111	No operation

37.15.9.3.2 Comparator and Time Base Selection

TBSA and TBSB fields (4-bit wide each) are used to configure the type of the comparator and the time bases used for match or capture (see the following tables). TBSA can also be used to control the Output Buffer Enable signal (See [Pin Control Registers](#)).

Table 37-396. Time Base Selection A - TBSA

TBSA[3] = 0	TBSA bit	2	1	0
	bitfield	Comparator selection	Capture selection	Match TB selection
	0	greater or equal	TCR1	TCR1
	1	equal-only	TCR2	TCR2
TBSA[3] = 1	action	2	1	0
	set OBE=1 ¹	0	0	0
	set OBE=0	0	0	1
	do nothing	1	1	1
	reserved	all other values		

1. Output Buffer Enable is not available on this chip.

Table 37-397. Time Base Selection B - TBSB

TBSB[3] = 0	TBSB bit	2	1	0
	bitfield	Comparator selection	Capture selection	Match TB selection
	0	greater or equal	TCR1	TCR1
	1	equal-only	TCR2	TCR2
TBSB[3] = 1	action	2	1	0
	do nothing	1	1	1
	reserved	all other values		

37.15.9.3.3 Transition Detection and Pin Action Control

IPACA/B and OPACA/B fields are used to configure transition detection sensitivity (for the channel input signal) or output pin action control (for the channel output signal), as defined in the table below. IPACA and IPACB have the same format, where IPACA is related to Match A and first transition detection, and IPACB to Match B and second transition detection. The same applies in analogue way to OPACA and OPACB.

For the output signal, configuring OPAC registers does not change the current signal state, but defines the action to be done when a match or transition detection occurs. See [Match Recognition](#) and [Channel Modes on Output Signal Generation](#) for more information.

IPACA/B=1xx also enables assertion of MRLA/B during Time Slot Transition. See [Match Recognition](#).

Table 37-398. Input and Output Pin Action Control - IPACA/B and OPACA/B

value	IPAC meaning	OPAC meaning
000	Do not detect transitions	Do not change output signal
001	Detect rising edge only	Match ¹ sets output signal high
010	Detect falling edge only	Match ¹ sets output signal low
011	Detect both edges	Match ¹ toggles output signal
100	Detect input signal = 0 on Match ¹	Transition detection sets output signal low
101	Detect input signal = 1 on Match ¹	Transition detection sets output signal high
110	reserved	Transition detection toggles output signal
111	do not change IPAC	do not change OPAC

1. Match A is used for IPACA/OPACA, and Match B for IPACB/OPACB.

37.15.9.3.4 Immediate Pin State Control

It is possible to change output signal state immediately by using PSC (2 bits) and PSCS (1 bit) fields.

Table 37-399. Immediate Pin State Control - PSC and PSCS

PSC	PSCS	Meaning
00	0	set signal as specified by OPACA (see Transition Detection and Pin Action Control)
00	1	set signal as specified by OPACB (see Transition Detection and Pin Action Control)
01	x	set signal high
10	x	set signal low
11	x	don't change signal state

37.15.9.3.5 Write Channel Match and UDCM Registers

Match registers can have their values changed using ERWA and ERWB fields (1 bit each). They also set their respective MRLE register (see [Match Recognition](#)).

ERWA can also be used to program the UDCM register (see [UDCM - User Defined Channel Mode](#)). The field CMW selects where the contents of ERTA is copied when ERWA is active (see the following table).

Table 37-400. Write MatchA/B - ERWA/B

Field	CM W	Value	Action
ERWA	1	0	write ERTA value in MatchA. Enable matches for MatchA register (MRLEA=1)
	0	0	write ERTA value into UDCM
	1	1	don't change UDCM, MatchA and MRLEA
	0	1	reserved
ERWB	1	0	write ERTB value in MatchB. Enable matches for MatchB register (set MRLEB=1)
	1	1	don't change MatchB and MRLEB
	0		
	0	0	reserved

If ERTA or ERTB is a destination of an ALU operation and, at the same time, the respective ERWA/B field is active, the new ERTA value is the one written into the MatchA/B register or the UDCM register.

37.15.9.3.6 Clear Transition/Match Event Registers

Flags MRLA, MRLB, TDLA and TDLB (see [ER - Event Registers](#)) indicate the state of matches and transitions detected in the selected channel, and it is possible to clear those flags using the microcode fields MRLA, MRLB (1 bit each) and TDL (1 or 2 bits, depending on the format). The flags cleared by these microcode fields are the actual channel flags, and also the ones sampled into the branch logic.

TDL can be one or two bits wide, depending on the microinstruction format (see [Microinstruction formats](#)). Two-bit TDL allows independent clearing of TDLA and/or TDLB. [Table 37-402](#) defines the two-bit TDL field.

Table 37-401. Clear Transition/Match Event Registers - MRLA/B, TDL

Field	Meaning
MRLA	0 = clear MRLA event register, 1 = don't change
MRLB	0 = clear MRLB event register, 1 = don't change
TDL (1 bit)	0 = clear TDLA and TDLB flags, 1 = don't change

Table 37-402. Independent TDLA/B clear - two-bit TDL

value	Meaning
0 0	clear TDLA
0 1	clear TDLB
1 0	clear both TDLA and TDLB
1 1	do not clear TDLA or TDLB

37.15.9.3.7 Disable Matches

Microcode field MRLE (1 bit) allows disabling matches on channel selected by CHAN register, for both MatchA and MatchB registers, by clearing their respective MRLE bits. Matches can be enabled for each Match register using ERWA and ERWB fields (see [Write Channel Match and UDCM Registers](#)).

Table 37-403. Disable Matches - MRLE

MRLE	Meaning
0	Disable matches for Match A and Match B
1	don't change match enabling

Some instruction formats have a two-bit MRLE field (see [Microinstruction formats](#)) which allows independent disabling of Matches 1 and 2, as shown in the following table.

Table 37-404. Two-bit MRLE

MRLE	Meaning
0 0	Disable Match A (clear MRLEA)
0 1	Disable both Matches (clear MRLEA and MRLEB)
1 0	Disable Match B (clear MRLEB)
1 1	nop

37.15.9.3.8 Disable Match and Transition Service Requests

Microcode field MTD (2 bits) disables match and transition service requests for the selected channel. MTD does not disable Link Service Request and Host Service Request. MTD sets or resets register SRI (for more details see [SRI - Match/Transition Service Request Inhibit Latch](#)) and TCCEA (see [TCCEA - Transition Continuous Capture Enable](#)) .

Table 37-405. Disable Match and Transition Service Request - MTD

MTD	Action on SRI	Action on TCCEA
00	SRI = 0: enable service requests for match and transition	TCCEA = 0: disable transition captures ¹ when TDLA = 1
01	SRI = 1: disable service requests for match and transition	
10	SRI = 1: disable service requests for match and transition	TCCEA = 1: enable transition captures ² when TDLA = 1
11	don't change	

1. Disables only captures on transition events specified by IPACA.

2. Enables only captures into CaptureA register, on transition events specified by IPACA.

37.15.9.3.9 Predefined Channel Modes

Microcode PDCM field (4 bits) defines the channel mode (see [Channel Modes](#)).

PDCM coding is shown in the table below. Note that PDCM bit 0 selects between Single Transition (PDCM[0]=0) and Double Transition (PDCM[0]=1) predefined modes.

PDCM is also used to select the user-defined channel mode, defined by the channel register UDCM (see [UDCM - User Defined Channel Mode](#)).

Table 37-406. Predefined Channel Modes

PDCM	Channel mode
0000	em_b_st
0001	em_b_dt
0010	em_nb_st
0011	em_nb_dt
0100	m2_st
0101	m2_dt
0110	bm_st
0111	bm_dt
1000	m2_o_st
1001	m2_o_dt
1010	user-defined channel mode
1011	reserved
1100	sm_st
1101	sm_dt
1110	sm_st_e
1111	keep current channel mode

37.15.9.3.10 Channel Interrupt and Data Transfer Requests

Microcode can issue Interrupt Requests, Data Transfer Requests and Global Exception through CIRC field. For more information see [Interrupts and data transfer requests](#).

Table 37-407. Channel and Data Transfer Requests - CIRC

CIRC	Meaning
000	Channel Interrupt Request from selected channel
001	Data Transfer Request from selected channel
010	Channel Interrupt and Data Transfer requests from selected channel
011	Channel Interrupt and Data Transfer requests from serviced channel
100	Channel Interrupt Request from service channel

Table continues on the next page...

Table 37-407. Channel and Data Transfer Requests - CIRC (continued)

CIRC	Meaning
101	Data Transfer Request from service channel
110	Global Exception
111	don't request interrupt

37.15.9.3.11 Clear Link Service Request

Microcode LSR field (1 bit) is used to clear the Link Service Request flag of the serviced channel (may not be the one selected by CHAN). The LSR branch condition is always cleared, but not the Link Service Request, if another channel link was received by the serviced channel during the executing thread. See [Channel Link](#) for more information.

Table 37-408. Link Service Request Negation Control - LSR

LSR	Meaning
0	clear Link Service Request (flag LSR)
1	don't change

37.15.9.4 Flow Control Microoperations

eTPU has *jump* and *call* microoperations to change microcode flow. Besides, eTPU has *dispatch jump* and *dispatch call* that can be used to implement a jump table. In *call* (or *dispatch call*) microoperation, the return address is saved in RAR register. If nested sub-routine calls are necessary, return address values have to be saved in a stack, usually implemented with DIOB register.

Flow Control microoperations are also provided to repeat a given microinstruction, to finish the current thread execution, and to halt the microengine.

37.15.9.4.1 Ending Current Thread - END

Microcode END field (1 bit) finishes current thread and allows other channels to be serviced. If END field is 0, the current instruction is completed and the thread is finished. END = 1 has no effect, and the next microinstruction is executed Any MDU operation (see [MAC and Divide Unit \(MDU\)](#)) that could be still pending when the thread is finished is left incomplete. END also releases any semaphore locked by the Engine.

37.15.9.4.2 Branch Operations

Branch operations can be **jump** or **call**. The target address of *jump* or *call* microoperations is always immediate and absolute. Branch microoperation is affected by FLS field (refer to [Flush Pipeline](#)).

37.15.9.4.2.1 Selecting Jump or Call Microoperations

The only difference between *jump* and *call* microoperations is that when a *call* is executed the value of PC or PC+1 (depending on flush, see [Flush Pipeline](#)) is saved in RAR register. The microcode field J/C (1 bit) selects whether *jump* or a *call* is executed, according to the following table.

Table 37-409. Jump / Call Selection - J/C

J/C	Meaning
0	jump
1	call

37.15.9.4.2.2 Branch Target Address

Microcode BAF field (14 bits) indicates the absolute address of a *jump/call target*.

37.15.9.4.2.3 Conditional/unconditional branch

Jump and *call* can be conditional or unconditional, depending on the BCC (6 bits) and BCF (1 bit) fields, as shown in [Table 37-410](#) and [Table 37-411](#). BCF determines whether branch is taken when condition specified by BCC is true or false. When a branch condition uses the channel flags, the channel context is related to the channel number written in CHAN register.

Table 37-410. Branch Condition Inversion - BCF

BCF	Meaning
0	branch if condition determined by BCC is false
1	branch if condition determined by BCC is true

Table 37-411. Branch Condition Selection - BCC

BCC	Meaning	BCC	Meaning
001110	Flag 0	001111	Flag 1
100000	V ALU flag	110000	PSS channel flag
100001	N ALU flag	110001	PRSS channel flag

Table continues on the next page...

Table 37-411. Branch Condition Selection - BCC (continued)

BCC	Meaning	BCC	Meaning
100010	C ALU flag	110010	"Less Than" ALU flag combination (signed) ¹
100011	Z ALU flag	110011	"Lower or Equal" ALU flag combination (unsigned) ²
100100	MV MDU flag	110100	P[24]
100101	MN MDU flag	110101	P[25]
100110	MC MDU flag	110110	P[26]
100111	MZ MDU flag	110111	P[27]
101000	TDLA channel flag	111000	P[28]
101001	TDLB channel flag	111001	P[29]
101010	MRLA channel flag	111010	P[30]
101011	MRLB channel flag	111011	P[31]
101100	LSR channel flag	111100	PSTO channel flag
101101	MB flag MDU flag	111101	PSTI channel flag
101110	FM[1] channel flag	111110	SMLCK semaphore flag
101111	FM[0] channel flag	111111	false
all other values reserved			

1. "less than" is a signed comparison, equal to the xor between ALU flags V and N; e.g., $0 < 0xFFFFFFFF$ tests as false ($0 < -1$).
2. "lower equal" is an unsigned comparison, equal to Z or C; e.g., $0 < 0xFFFFFFFF$ tests as true.

37.15.9.4.3 Dispatch Microoperation

Dispatch microoperation is an unconditional branch where the target address is always $PC + P[31:24]$ (unsigned). Dispatch is affected by FLS field (refer to [Flush Pipeline](#)). Dispatch microoperation is defined by R/D field (2 bits, see the following table). Field R/D can also be used to define return from sub-routine (see [Return from subroutine](#)).

Table 37-412. Return and Dispatch - R/D

R/D	Meaning
00	return from subroutine (see Return from subroutine)
01	dispatch jump
10	dispatch call
11	don't change microinstruction flow

37.15.9.4.4 Return from subroutine

When a subroutine *call* or a *dispatch call* microoperation is executed, the return address is saved in RAR register. To return from a subroutine a microoperation is available to load the contents of RAR register back to the PC. Fields R/D (2 bits) or RTN (1 bit, see the table below) can be used to return from subroutine. For R/D field, see [Table 37-412](#).

Return from subroutine microoperation is affected by FLS (see [Flush Pipeline](#)) when field R/D is used. Return execution through RTN always flushes the pipeline.

Table 37-413. Return from subroutine - RTN

RTN	Meaning
0	Return with pipeline flush
1	Do not return

37.15.9.4.5 Flush Pipeline

When a branch, dispatch or subroutine return microoperation is executed, the next microinstruction can be executed unconditionally before the flow change takes effect, since microengine has a three-stage pipeline. Executing the next microinstruction after a branch maximizes execution performance. This feature is controlled by field FLS (1 bit, see the table below). When FLS=0 the pipeline is flushed, so the next microinstruction placed after a branch is decoded as NOP if the branch is taken. If FLS=1, the microinstruction placed after the branch is executed, either if the branch is taken or not, as shown in the figure below.

Flush also controls which value is stored in RAR in a call: in case of no flush, it is the address of the branch/dispatch instruction + 2, even if RAR is the ALU destination of the instruction after the call; in case of a flush, it is the address of the instruction following branch/dispatch.

If a branch with no flush is followed by another branch with no flush, the instructions are executed in the following order:

1. first branch
2. second branch
3. first branch's destination instruction
4. second branch's destination instruction, and the flow proceeds normally from then on

The destination of the first branch must not be another flow changing instruction (branch, return or dispatch). Similar flows apply when returns or dispatches are used instead of branches. This scheme can be used to implement quick table look-ups with a dispatch replacing the first branch, for instance.

Table 37-414. Flush Pipeline - FLS

FLS	Meaning
0	flush pipeline when jump / call / dispatch jump / dispatch call / return is executed
1	do not flush pipeline when jump / call / dispatch jump / dispatch call / return is executed

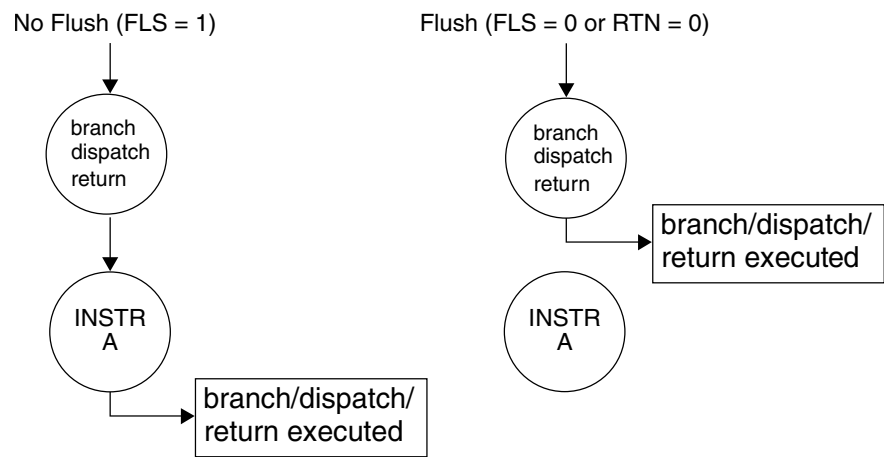


Figure 37-331. Flush Pipeline

37.15.9.4.6 HALT microinstruction

HALT is a microinstruction provided to implement software breakpoints (see [Software breakpoints](#)). Note that HALT is coded as a microinstruction format, not a field (see [Microinstruction formats](#)). The execution of this instruction puts the microengine in halt state. For more information about the implications of microengine halt state, see [Microengine Halt State](#) . HALT is valid only if software breakpoints are enabled at the Debug interface . If software breakpoints are not enabled, HALT executes as a NOP and is treated as an Illegal Instruction (see [Illegal instructions](#)).

37.15.9.4.7 NOP microinstruction

There is not a unique microinstruction with an assigned opcode to do No Operation. NOP microinstruction is achieved through any of the formats shown on [Microinstruction formats](#) where the user can assign to each individual field the corresponding value for "No Operation". However, to prevent future impacts of instruction changes on object code compatibility, the instruction value 0x4FFFFFFF should always be used for NOP.

37.15.9.5 Illegal instructions

An instruction is considered illegal if any reserved field value is used, including when the fields marked rsv in the instruction formats (see [Microinstruction formats](#)) are assigned value 0. A HALT instruction is exceptionally considered illegal when executed with software breakpoints disabled (see [HALT microinstruction](#)). Global Exception may be issued up to two microcycles after instruction fetch. The execution results of an illegal instruction on the microengine, channel logic or host interface are unspecified, except for the HALT case.

If the microengine decodes an illegal instruction, the following actions are taken:

- A Global Exception is issued.
- The ILF1/2 flags in [eTPU Module Configuration Register \(eTPU_SCR_MCR\)](#) are set to indicate this occurrence to the Host.
- A breakpoint occurs, if NDEDI is present and configured to do so (see the NDEDI details for more information).

37.15.9.6 Microinstruction Parallelism Issues

This section clarifies parallelism issues that arise when two non-commutative microoperations appear in the same microinstruction.

37.15.9.6.1 ALU Operations and Read Match Registers

ALU operations have only one destination register, but there is one case where source selection determines destination: read Match register in ERTA and ERTB registers. In this case if ALU destination is ERTA or ERTB a conflict arises. The ALU destination value overwrites the value read from the match registers.

37.15.9.6.2 ALU and SDM Operations

P and DIOB registers can be selected as destination by both ALU and SDM (read) microoperations in the same microinstruction. Since P and DIOB update from SDM data happens after P and DIOB update for ALU/MDU microoperations, the data read from SDM remains in P or DIOB after an operation when one of them is specified as destination for both ALU and SDM microoperations. In this case, the value loaded into P or DIOB is the one read from SDM. However, the ALU operation is executed and its flags are updated accordingly.

When P or DIOB is destination of an SDM read and also an ALU source at the same microinstruction, the value before the read is used for the ALU operation.

If DIOB is the ALU destination and P is loaded from SDM or vice-versa, no conflict occurs, and the result is the same as if operations occurred separately.

All the above also applies to Zero SDM operations.

When using P or DIOB as destination for ALU operations and also as source for a SDM write operation, the data written in SDM is the one calculated by ALU, which means it is possible to calculate a value and write it in an SDM address using only one microinstruction.

The old value of DIOB or the old value minus 4 (pre-decrement) is always used when DIOB is selected as address (indirect address mode), no matter if DIOB is selected as destination of either the SDM or ALU. For the value loaded into DIOB, refer to the following table.

Table 37-415. DIOB load from SDM and ALU

DIOB selected as SDM read destination?	DIOB selected as ALU destination?	DIOB load value
no	no	DIOB, --DIOB (pre-decrement), or DIOB++ (post-increment)
yes	no	SDM read data (post-inc and pre-dec ignored)
yes	yes	SDM read data (post-inc, pre-dec and ALU result ignored)
no	yes	ALU result (post-inc and pre-dec ignored)

When CHAN is the destination of an ALU operation in parallel with a channel-relative SDM access, the CHAN value used to calculate the address is the one before the ALU operation.

37.15.9.6.3 ERTA/B as ALU destination and ERWA/B

The value in ERTA and ERTB registers can be written in match registers of the selected channel by using fields ERWA and ERWB ([Write Channel Match and UDCM Registers](#)). If, at the same microinstruction, ERTA or ERTB is destination of an ALU/MDU microoperation, the value written in Match registers is the ALU/MDU result.

The same applies to UDCM when ERTA is the destination of an ALU operation and instruction fields ERWA and CMW are active .

If an ALU operation occurs in parallel with ERWA/B but ERTA/B are not the destination of an ALU/MDU operation, then UDCM and MatchA/B receives the ERTA/B value.

37.15.9.6.4 ERWA/B and MRLE

ERWA/B automatically sets the MRLEA/B channel latch, respectively (see [Write Channel Match and UDCM Registers](#)). Microinstruction fields ERWA/B independently sets MRLEA/B channel flags, regardless of MRLE.

37.15.9.6.5 CHAN assignment, Read Match and ERWA/B

When CHAN is a destination of an ALU operation it causes a read of the CaptureA/B register values into ERTA/B. The Capture registers loaded into ERTA/B are selected by the new CHAN value. The value of the CaptureA/B registers overwrites any read-match commanded simultaneously.

If CHAN assignment happens with an ERWA/B operation in the same instruction, the updated Match register(s) belong to the new selected channel.

37.15.9.6.6 Read Match and ERWA/B

If a read match operation is executed with ERWA/B in the same microinstruction, the MatchA/B registers receive the old values of ERTA/B, and the ERTA/B registers receive the old MatchA/B values simultaneously, i.e.: ERTA/B and MatchA/B swap their values.

If ERTA/B is the destination of an ALU operation at the same instruction, MatchA/B gets the ALU result (see [ERTA/B as ALU destination and ERWA/B](#)), but the ERTA/B not being written still receives the old MatchA/B values.

Note

Read match, ERWA/B and CHAN assignment can be active at the same instruction. Combining rules [CHAN assignment, Read Match and ERWA/B](#) and [Read Match and ERWA/B](#), the result is: ERTA/B receives the CaptureA/B values of the new CHAN value, and MatchA/B of the new channel receives the old ERTA/B value(s).

37.15.9.6.7 Stack accesses and ALU operations

Post-increment is ignored in a stack operation (field STC) if DIOB is loaded from SDM: DIOB keeps the value read from SDM. Pre-decrement is ignored in a stack operation (field STC) if DIOB is destination of an ALU operation, for DIOB load value, but not for DIOB as address. Post-increment/pre-decrement remains valid in all other situations. These rules can be summarized in the following equivalent C code, and in [Table 37-415](#).

```
DIOB = *DIOB[15:2];      // read without posinc/predec
*DIOB[15:2] = DIOB;      // write without posinc/predec
DIOB = * (--DIOB[15:2]); // read with predec
DIOB = *DIOB[15:2];      // read with posinc (ignored)
*(--DIOB[15:2]) = DIOB;  // write with predec
*DIOB[15:2]++ = DIOB;    // write with posinc (value written is before
increment)
```

37.15.9.6.8 SRC and ALU/MDU operations

If operation SRC is active (field SRC = 0) and register SR is selected as destination of an ALU operation, the value of the ALU operation prevails over the shifted value.

The value of SR used as source in the ALU/MDU operation is the one before the shift.

37.15.9.6.9 Semaphore lock/free and SMLCK branch condition

When the SMLCK branch condition is tested at the same microinstruction of a semaphore lock or free, the condition is evaluated after the semaphore action (either free or lock) is taken.

37.15.9.6.10 Dispatch and SDM Read

When the most significant byte of P is read from SDM (read 8 msb bits or 32 bits) and a dispatch instruction is executed simultaneously, the dispatch target address is calculated upon the P value before the read.

37.15.9.6.11 CHAN Assignment, PSC/PSCS, and clear MRLEA/B, MRLA/B, TDLA/B

When clear MRLEs, MRLA/B or TDLs is done and a CHAN assignment is done at the same time, the flag selected by the old CHAN value is cleared in the channel, but the branch conditions receive the state of the flags selected by the new CHAN.

When a pin action is commanded through PCS/PSCS and a CHAN assignment is done simultaneously, the output signal affected is selected with the old CHAN value.

37.15.9.7 Microinstruction formats

format	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																											
A1	0 0 0			IMM[15:13]		IMM[7:2]							IMM[23:16]										IMM[12]	RTN	IMM[11:9]			T2D			IMM[8]		0 0																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																										
A2																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																											
A3																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																											
A4																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																											
B1	1	0	0	ALUOP[4]	CCSV	SHF	CIN	BINV	T4BBS			RW	T4ABS										T2ABD			SRC	P/D	AS/CE	FLC[1:0]	[0:1]MMW							AB SE	AB DE	1	0 1 0																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																			
B2	1	END																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																									
B3	0 0 0																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																										
B4	0		0																																							1	CCSV	CIN	BINV	T4BBS	RW	FL	rsv	TDL	TBSA										TBSB			LSR	PSC	MRLA	ERWA	MRLB	ERWB	ABSE	ABDE	CCS	MRLE	PSCS																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																	
B5	0	0	1																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																								
B6	0	0	1																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																								
B7	0	1	1	END	SHF	END	SHF	END	SHF	END	SHF	END	SHF	END	SHF	END	SHF	END	SHF	END	SHF	END	SHF	END	SHF	END	SHF	END	SHF	END	SHF	END	SHF	END	SHF	END	SHF	END	SHF	END	SHF	END																																	SHF	END	SHF	END	SHF	END	SHF	END	SHF	END	SHF	END	SHF	END	SHF	END	SHF	END	SHF	END	SHF	END	SHF	END	SHF	END	SHF	END	SHF	END	SHF	END	SHF	END	SHF	END	SHF	END	SHF	END	SHF	END	SHF	END	SHF	END	SHF	END	SHF	END	SHF	END	SHF	END	SHF	END	SHF	END	SHF	END	SHF	END	SHF	END	SHF	END	SHF	END	SHF	END	SHF	END	SHF	END	SHF	END	SHF	END	SHF	END	SHF	END	SHF	END	SHF	END	SHF	END	SHF	END	SHF	END	SHF	END	SHF	END	SHF	END	SHF	END	SHF	END	SHF	END	SHF	END	SHF	END	SHF	END	SHF	END	SHF	END	SHF	END	SHF	END	SHF	END	SHF	END	SHF	END	SHF	END	SHF	END	SHF	END	SHF	END	SHF	END	SHF	END	SHF	END	SHF	END	SHF	END	SHF	END	SHF	END	SHF	END	SHF	END	SHF	END	SHF	END	SHF	END	SHF	END	SHF	END	SHF	END	SHF	END	SHF	END	SHF	END	SHF	END	SHF	END	SHF	END	SHF	END	SHF	END	SHF	END	SHF	END	SHF	END	SHF	END	SHF	END	SHF	END	SHF	END	SHF	END	SHF	END	SHF	END	SHF	END	SHF	END	SHF	END	SHF	END	SHF	END	SHF	END	SHF	END	SHF	END	SHF	END	SHF	END	SHF	END	SHF	END	SHF	END	SHF	END	SHF	END	SHF	END	SHF	END	SHF	END	SHF	END	SHF	END	SHF	END	SHF	END	SHF	END	SHF	END	SHF	END	SHF	END	SHF	END	SHF	END	SHF	END	SHF	END	SHF	END	SHF	END	SHF	END	SHF	END	SHF	END	SHF	END	SHF	END	SHF	END	SHF	END	SHF	END	SHF	END	SHF	END	SHF	END	SHF	END	SHF	END	SHF	END	SHF	END	SHF	END	SHF	END	SHF	END	SHF	END	SHF	END	SHF	END	SHF	END	SHF	END	SHF	END	SHF	END	SHF	END	SHF	END	SHF	END	SHF	END	SHF	END	SHF	END	SHF	END	SHF	END	SHF	END	SHF	END	SHF	END	SHF	END	SHF	END	SHF	END	SHF	END	SHF	END	SHF	END	SHF	END	SHF	END	SHF	END	SHF	END	SHF	END	SHF	END	SHF	END	SHF	END	SHF	END	SHF	END	SHF	END	SHF	END	SHF	END	SHF	END	SHF	END	SHF	END	SHF	END	SHF	END	SHF	END	SHF	END	SHF	END	SHF	END	SHF	END	SHF	END	SHF	END	SHF	END	SHF	END	SHF	END	SHF	END	SHF	END	SHF	END	SHF	END	SHF	END	SHF	END	SHF	END	SHF	END	SHF	END	SHF	END	SHF	END	SHF	END	SHF	END	SHF	END	SHF	END	SHF	END	SHF	END	SHF	END	SHF	END	SHF	END	SHF	END	SHF	END	SHF	END	SHF	END	SHF	END	SHF	END	SHF	END	SHF	END	SHF	END	SHF	END	SHF	END	SHF	END	SHF	END	SHF	END	SHF	END	SHF	END	SHF	END	SHF	END	SHF	END	SHF	END	SHF	END	SHF	END	SHF	END	SHF	END	SHF	END	SHF	END	SHF	END	SHF	END	SHF	END	SHF	END	SHF	END	SHF	END	SHF	END	SHF	END	SHF	END	SHF	END	SHF	END	SHF	END	SHF	END	SHF	END	SHF	END	SHF	END	SHF	END	SHF	END	SHF	END	SHF	END	SHF	END	SHF	END	SHF	END	SHF	END	SHF	END	SHF	END	SHF	END	SHF	END	SHF	END	SHF	END	SHF	END	SHF	END	SHF	END	SHF	END	SHF	END	SHF	END	SHF	END	SHF	END	SHF	END	SHF	END	SHF	END	SHF	END	SHF	END	SHF	END	SHF	END	SHF	END	SHF	END	SHF	END	SHF	END	SHF	END	SHF	END	SHF	END	SHF	END	SHF	END	SHF	END	SHF	END	SHF	END	SHF	END	SHF	END	SHF	END	SHF	END	SHF	END	SHF	END	SHF	END	SHF	END	SHF	END	SHF	END	SHF	END	SHF	END	SHF	END	SHF	END	SHF	END	SHF	END	SHF	END	SHF	END	SHF	END	SHF	END	SHF	END	SHF	END	SHF	END	SHF	END	SHF	END	SHF	END	SHF	END	SHF	END	SHF	END	SHF	END	SHF	END	SHF	END	SHF	END	SHF	END	SHF	END	SHF	END	SHF	END	SHF	END	SHF	END	SHF	END	SHF	END	SHF	END	SHF	END	SHF	END	SHF	END	SHF	END	SHF	END	SHF	END	SHF	END	SHF	END	SHF	END	SHF	END	SHF	END	SHF	END	SHF	END	SHF	END	SHF	END	SHF	END	SHF	END	SHF	END	SHF	END	SHF	END	SHF	END	SHF	END	SHF	END	SHF	END	SHF	END	SHF	END	SHF	END	SHF	END	SHF	END	SHF	END	SHF	END	SHF	END	SHF	END	SHF	END	SHF	END	SHF	END	SHF	END	SHF	END	SHF

ALU Operations

Channel Control/Config Operations

RAM Operations

Flow Control Operations

NOTES:

¹ format E4 is deprecated; the same result can be obtained with format E1 with field STC=11

Figure 37-332. Microinstruction formats

All bits marked "rsvd" are reserved, and must be coded as 1.

37.15.10 Shared Subsystems support

Shared Subsystems support allows eTPU to control other modules, either independently or sharing control with the host. A notable example of shared subsystem is the Reaction Module (see the Reaction Module chapter for details), which is connected to the eTPU as shown in the figure below. The host CPU accesses the Reaction Module through the eTPU, which can also access the same registers.

When the SSMAE field in [eTPU Channel n Configuration Register for Engine 1 \(eTPU_CCCR_CnCR_ENG1\)](#) or [eTPU Channel n Configuration Register for Engine 2 \(eTPU_CCCR_CnCR_ENG2\)](#) is 1, the eTPU channel's threads can control shared subsystems by accessing their registers, which are mapped onto the higher 16 Kbytes of the microcode data address space (byte addresses/DIOB values 0x4000 to 0x7FFF, equivalent to parameters 0x1000 to 0x1FFF), as shown in [Figure 37-334](#). The shared subsystem registers are accessible by microcode only with the indirect address mode (see [SDM Addressing Modes](#)).

When the SSMAE field in [eTPU Channel n Configuration Register for Engine 1 \(eTPU_CCCR_CnCR_ENG1\)](#) or [eTPU Channel n Configuration Register for Engine 2 \(eTPU_CCCR_CnCR_ENG2\)](#) is 0, the channel's threads can access only the SDM, mirrored throughout the whole DIOB address range.

The same Shared Subsystem register address space is accessible by the host through the Shared Subsystem Area within the eTPU memory map (see [Figure 37-334](#) and [Memory map](#)), allowing shared control and/or supervision between the host and the eTPU. The arbitration mechanism between host and eTPU accesses is the same used for SDM accesses (see [SDM arbitration](#)). The CDC (see [Coherent Dual-parameter Controller \(CDC\)](#)) cannot be used for coherent transfers between SDM and the Shared Subsystem Area, or within the Shared Subsystem Area itself.

Note

The detailed register map within the Shared Subsystem Area depends on the subsystem module.

Bus errors coming from the Shared Subsystem cause a Global Exception and are flagged by the MCR[SSAERR] bit on microengine accesses only. In the case of host access, the bus error only propagates to the slave bus, which also issues a bus error.

A Shared Subsystem may be divided into submodules or channels, hereafter called **Shared Subsystem Units** (or **Shared Units** for short), which can be individually controlled by an eTPU function (see [Functions and threads](#))

A Shared Unit is logically linked with an eTPU channel through the eTPU_SSIR_ETPUSSSIR_n configuration³⁸. There is one eTPU_SSIR_ETPUSSSIR register for each Shared Unit to configure its interface with the eTPU (see [eTPU Shared](#)

[Subsystem n Integration Register \(eTPU_SSIR_ETPUSSSIR \$n\$ \)](#)). The Shared Unit uses the Link Service Request (LSR) mechanism to request service to the eTPU. The Link Service Request is targeted to the channel/engine configured in [LENG](#) and [LCHAN](#). The function assigned to this channel must be written to service the Shared Unit. The Shared Unit link service request may have the same source as an interrupt to the host.

Note

Shared Unit link service requests are level-sensitive to the eTPU. If it comes from an interrupt source, the link service request stays asserted as long as the interrupt is asserted and enabled, so the interrupt request must be cleared before the clear of the LSR flag in the link service request thread.

[LENG](#) can also be used by the host to disable Link Service Requests to the eTPU. The eTPU microcode can only disable LSRs from the Shared Unit through configuration specific to the Shared Unit itself.

Note that more than one Shared Unit can generate LSRs to the same eTPU channel, in which case the LSR source can be determined by microcode through the Shared Unit status register(s). An eTPU channel configured as a Shared Unit LSR target can also receive Link Service Requests from other eTPU channels or Shared Units. The same LSR service and flag behavior found in links between eTPU channels applies to Shared Unit links (see [Channel Link](#)).

The table below summarizes the mapping of Reaction Module resources into Shared Subsystem Support.

38. However, in the case of Reaction Module, the eTPU_SSIR_ETPUSSSIR n configuration does not determine which eTPU channel output enables the Reaction Channel modulation; this is defined in the Reaction Module configuration.

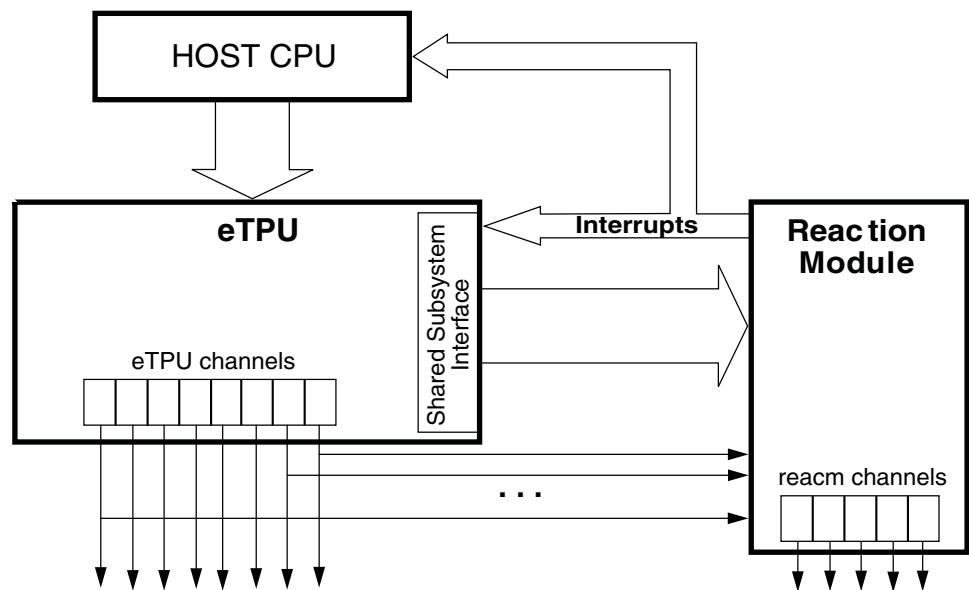


Figure 37-333. Shared Subsystem block diagram example: Reaction Module

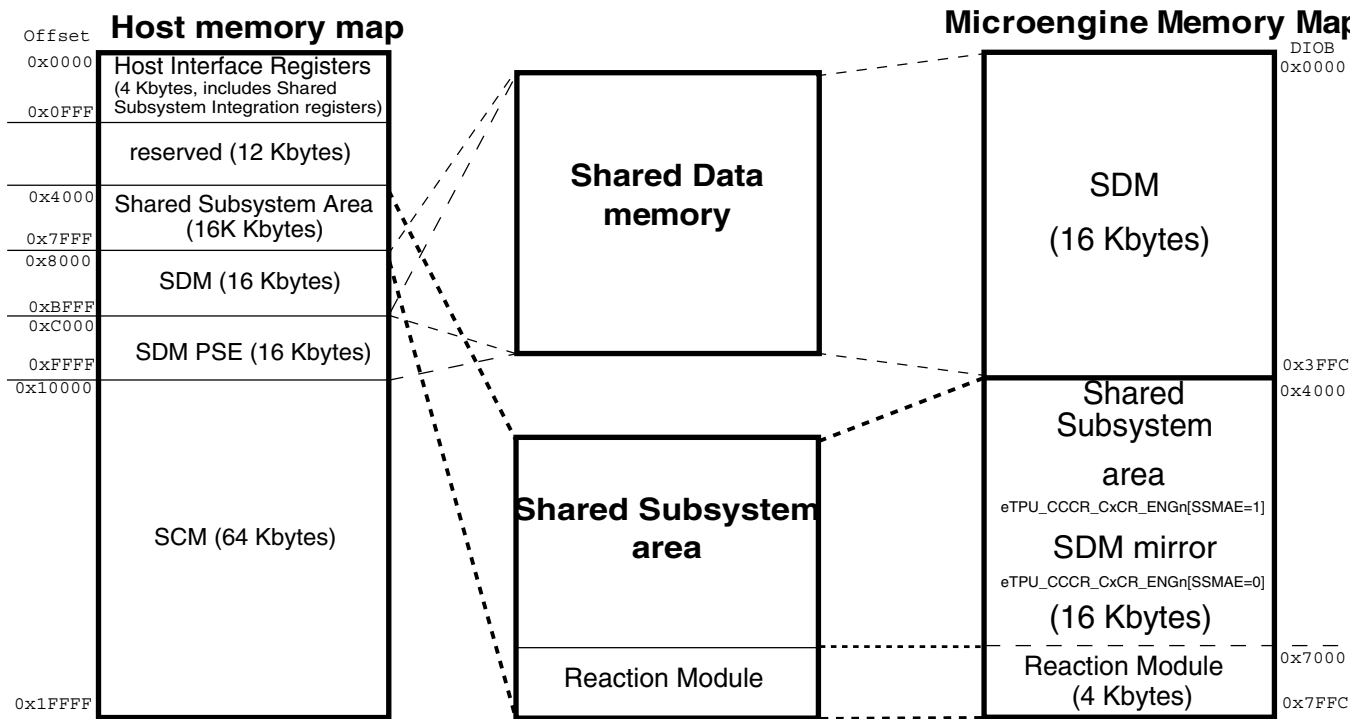


Figure 37-334. Shared Subsystem memory mapping

Table 37-416. Reaction Module Shared Subsystem support

Reaction Module resource	Host mapping	eTPU/microengine mapping
Register access	eTPU Base + 0x7000	DIOB address 0x7000 (indirect address mode)

Table continues on the next page...

Table 37-416. Reaction Module Shared Subsystem support (continued)

Reaction Module resource	Host mapping	eTPU/microengine mapping
Interrupts	See Interrupts chapter	Link Service Requests mapped through Subsystem Integration registers eTPU_SSIR_ETPUSSSIR _n
DMA Requests	See eDMA chapter	Not applicable

37.15.11 Development support features

Following sections describe several features available to support development. Most debug features are accessible through a separate debug bus, and are not available through registers in the standard eTPU memory map. The details of the access to this interface are MCU-dependent (see Debug Interface chapter). The microengines can also be put in halt state by a device-level debug request.

NOTE

The behavior of the debug support logic is not specified in the event of a non-correctable memory error on either SCM or SDM (see [Memory error support](#)).

37.15.11.1 Nexus Class 3 Support

NDEDI is an IP block designed to support Nexus functionality for the eTPU. It provides a Nexus class 3 or class 1 debug interface, depending on the MCU (see Configuration section). Nexus is a development support external interface defined by the **IEEE standard ISTO 5001-1999**. More details about it can be found in the NDEDI details.

Some of the next subsections describe debug features provided by the eTPU microengine debug support combined with the NDEDI block.

37.15.11.2 Microengine Halt State

Halt is a microengine state where it suspends execution during a thread, or does not start executing a scheduled thread from idle state. While Idle State is entered from END execution without any other scheduled thread, microengine enters Halt State by any of the following events:

- execution of the HALT microinstruction (software breakpoint).

- external halt request through the Debug Interface (includes Nexus breakpoint request via EVTI input pin - see [Nexus Class 3 Support](#)).
- the other Engine enters halt state and they are configured to halt simultaneously (bit HTWIN is asserted via Nexus Interface).
- System level device debug request assertion and NDEDI register NDEDIETPUx_DC field CBI=1. If same register's field CBT=1, microengine halts at the next time-slot boundary, if CBT=0 it halts immediately. As a particular case, microengines come halted out of reset if device debug request is asserted, since CBI reset value is 1. Microengine does not execute out of reset, either in halt (device debug request asserted) or idle state (device debug request negated), but halt enables several other features (see below).
- occurrence of any of the hardware breakpoint conditions. See [Hardware breakpoints](#) for details.
- execution of a single-step microinstruction: microengine returns to halt state after executing a single microinstruction while in halt state. See [Single-step Execution](#) and [Forced microinstruction execution](#) for details.

When microengine enters halt state, it automatically triggers the following actions:

- suspends input signal sampling and filters (respective Engine channels only), if NDEDI_ENGINEx_DC[PINS] = 1 at the Debug Interface.
- releases the SDM arbitration for Host or CDC accesses, no matter if microengine was halted in the middle of a dual parameter (back-to-back) access.
- stops TCR1/2 clocks of the respective Engine, if signal ndedi_stop_tcr is asserted at the Debug Interface.
- if the other Engine is also in halt state or stopped, allows turning the [eTPU Module Configuration Register \(eTPU_SCR_MCR\)](#) VIS bit to 1.

If all halt conditions are cleared when VIS=1, microengine(s) keep on halt state until VIS=0, when it automatically exits halt state, except on single-step (see [Single-step Execution](#)), so that single-step execution is ignored while VIS=1.

MDU continues executing until it finishes any ongoing operation even if microengine is in halt state, except when the halted instruction is an END.

There are two kinds of halt state, depending on the previous microengine state when halted:

1. **halt_idle**, if the engine was not executing a thread when halted; the engine cannot leave **halt_idle** to fetch instructions, so one cannot single-step or follow a program flow; it can, however, execute forced instructions (see [Forced microinstruction execution](#)).
2. **halt_exec**, if the engine was executing a thread when halted. The engine can single-step and continue a program flow from **halt_exec**.

When microengine exits halt state, any dependable action is suspended and, if exiting **halt_exec**, the instruction pointed by the PC is fetched, while the instruction already fetched before halt is executed. Note that both the PC and the prefetched instructions can be modified during halt state, with a forced execution of a branch instruction (see [Forced microinstruction execution](#)).

37.15.11.3 Hardware breakpoints

Microengine can enter halt state through a command from the Debug Interface, configuring a hardware breakpoint. Hardware breakpoints can halt the microengine on specific conditions, listed below. These conditions depend on NDEDI configuration (see the NDEDI details for more information).

- CHAN register assignment (only by microcode, not by time slot transition).
- SDM read and/or write to a given address and/or write data. The breakpoint is always qualified by the SDM address, but the following variations are allowed:
 - break on write only, read only, or read-and-write.
 - break on higher-byte write data value, lower 24-bit write value, full word (32-bit) write value, or regardless of data. Break on read data is not supported.
- PC (program counter) value.
- Beginning of a thread with a Host Service Request pending.
- Beginning of a thread with a Link Service Request pending.
- Beginning of a thread with a Match Service Request pending.
- Beginning of a thread with a Transition Service Request pending.
- End of a thread.
- Illegal instruction execution.

All these conditions can also be qualified by the value of the CHAN register.

NOTE

Data breakpoints are not supported on Shared Subsystem Area accesses (see [Shared Subsystems support](#)).

Hardware breakpoints halt the microengine before the halted instruction is executed, except on data breakpoints. Data breakpoints allow the instruction that performs the data access to be executed before the microengine is halted.

Note

Data breakpoints in eTPU versions before eTPU2 halt the breakpoint instruction before its execution in all cases, including data breakpoints.

On any of these conditions, halt of one microengine does not depend on the halt of the other, unless the other engine is configured to do so, via Nexus Interface. Occurrence of any of these conditions halts the microengine, i.e., the conditions are logically ORed together, and they can be individually enabled.

While in halt state, the microengine can also execute any forced microinstruction not in the normal program flow (see [Forced microinstruction execution](#)) or, if in halt_exec, in single-step (see [Single-step Execution](#)).

There are situations when requests for stopping an engine, breakpoint and service can occur simultaneously. Breakpoint requests always prevails over a stop request ([eTPU Engine 1 Configuration Register \(eTPU_SCR_ECR_ENG1\)](#) or [eTPU Engine 2 Configuration Register \(eTPU_SCR_ECR_ENG2\)](#) bit MDIS=1 or device debug request=1). When the eTPU is idle: stop request prevails over Service Request if there is not a hardware breakpoint request; a hardware breakpoint request leads to debug mode immediately if there is no Service Request, and after TST if there is Service Request (regardless of stop requests). The rules above are summarized in the the following table, showing the destination state of the microengine in each situation.

Table 37-417. Breakpoint, stop and service requests resolution from idle

Breakpoint request	MDIS	Service request	Final state
no	0	0	Idle
no	1	0	Stop
no	0	1	TST
no	1	1	Stop
Yes	0	0	Halt_idle
Yes	1	0	Halt_idle
Yes	0	1	TST ¹
Yes	1	1	TST ¹

1. Breaks after TST, if signal `ndedi_sync_break` is still asserted.

When a thread is ending, it goes to Idle or TST only if there is neither a hardware breakpoint request nor a request to stop ([eTPU Engine 1 Configuration Register \(eTPU_SCR_ECR_ENG1\)](#) or [eTPU Engine 2 Configuration Register \(eTPU_SCR_ECR_ENG2\)](#) bit MDIS=1 or device debug request=1). When thread is ending and there are simultaneous hardware breakpoint and stop (MDIS=1 or device debug request=1) requests, hardware breakpoint prevails and the engine enters Debug mode (Halt_idle state).

If the engine entered Debug mode after a thread finished (Halt_idle state) and a "go" command comes from the debug interface, the engine state machine goes to Idle and the rules above apply. It means that if a "go" is issued in Halt_idle state with MDIS=1, the engine goes to Idle for one microcycle and then stops (if MDIS or device debug request keeps asserted and there is no other breakpoint request).

Note

Hardware breakpoint requests are ignored for the first microinstruction executed when microengine leaves halt.

37.15.11.4 Hardware Watchpoints

Debug Interface allows watchpoints on the same conditions available for hardware breakpoints (see [Hardware breakpoints](#)).

Note

Data breakpoints are not supported on Shared Subsystem Area accesses (see [Shared Subsystems support](#)).

37.15.11.5 Software breakpoints

A software breakpoint occurs when microengine executes a HALT microinstruction. Any number of software breakpoints can be set in code, usually replacing an active microinstruction.

Like any other microinstruction, HALT increments the PC and pre-fetches the next instruction. So, before the halt state is suspended, if the original program flow must be followed, the original instruction at the HALT address must be executed, regardless if the software breakpoint is removed (replacing HALT by the original microinstruction) or not. Follows the procedure to resume execution from a software breakpoint:

1. restore the original instruction in SCM (replace HALT).
2. force a jump with flush to the original instruction (see [Forced microinstruction execution](#)).
3. if the software breakpoint must be kept: single-step and replace the original instruction with a HALT.
4. let the flow continue, issuing a GO command (leaving halt state).

Special care must be taken if HALT is followed by another HALT, and the second HALT is removed when microengine was halted by the first one. In this case, replacing the second HALT with the original microinstruction is not enough to remove the second breakpoint, because the second HALT was already prefetched and would be executed anyway when halt was suspended. The debugger must also do a forced execution of unconditional branch with flush to the original microinstruction address. That will clear the pipeline, replacing the prefetched instruction with a NOP, and load PC with the address of the removed breakpoint. So, when halt state is suspended, the original microinstruction will be fetched while NOP is executed, and program flow continues normally from then on.

Note

A HALT instruction placed after a no-flushing branch, dispatch or return may be a problem from the debugger application standpoint: after the HALT is executed, the eTPU debug interface informs the address of the branch/dispatch/return destination, and the debugger application has no direct way to identify which HALT instruction was executed, if multiple HALTs lead to the same address. This can be solved if the debug support block (NDEDI) has a register holding the address of the last instruction executed, otherwise one should forbid non-flushed HALT instructions. See the NDEDI details.

Software breakpoint setting and removal is possible only with SCM RAM implementations or ROM implementations with SCM RAM emulation (see [SCM Emulation](#)). There is only one way of inserting software breakpoints into SCM RAM: programming VIS=1 in [eTPU Module Configuration Register \(eTPU_SCR_MCR\)](#), and then accessing SCM as an ordinary RAM from the slave bus. This can be done only if both Engines are halted or stopped.

37.15.11.6 Single-step Execution

When microengine is already in halt_exec state, it can run the next microinstruction in the normal program flow and get back to halt state. PC is incremented, or assigned the BAF value in a branch with satisfied condition. Note that the executed instruction was already prefetched in the instruction pipeline, and a new microinstruction is fetched during its execution. The prefetched instruction may be cleared during halt state by the forced execution of a branch with flush (see [Forced microinstruction execution](#)), making single-step execute a NOP instead of the next instruction in the program flow.

Single-step execution is controlled by the debug interface, and is a feature available from Nexus if eTPU is connected to the NDEDI block (see NDEDI details).

The single-step execution of a NOP instruction can be useful to control input signal sampling and filtering, if NDEDI_ENGINEx_DC[PINS] = 1 at the Debug Interface (see NDEDI details). Single-step does not happen if VIS=1.

37.15.11.7 Forced microinstruction execution

When microengine is already in halt state (either halt_idle or halt_exec), it can run forced microinstructions through the debug interface. This feature is available from Nexus if eTPU is connected to the NDEDI block (see NDEDI details). The microinstruction, specified by the user, is not fetched from SCM and comes directly from the debug interface. MDU start commands issued by forced instructions are executed, and the MDU runs the operation until the end, independently of the halt state. The microinstruction field END is ignored.

During forced execution of any instruction except Branches, Returns and Dispatches, the PC does not change, and the prefetched instruction in the pipeline is bypassed, but not discarded. When halt state is suspended, the prefetched instruction is executed and the instruction pointed by the PC is prefetched in parallel (two-stage pipeline).

Forced execution of a Branch, Dispatch or Return loads the PC with the BAF field (if branch condition is satisfied), PC+P or RAR, respectively. If branch condition is not satisfied, PC value stays unaltered. The flush control (field FLS) also works, so that a successful forced branch with flush replaces the prefetched instruction with a NOP. So, to clear the instruction pipeline during halt, all one has to do is an unconditional branch to the desired address with flush. HALT instructions must not be executed as forced.

Forced operations that depend on the serviced channel are unspecified when executed in halt_idle and must be avoided.

Note

Forced instructions must not be commanded when [eTPU Module Configuration Register \(eTPU_SCR_MCR\)](#) field VIS=1.

37.15.11.8 Microengine Register Access

eTPU provides no direct access to microengine and channel registers from the slave bus or any other interface. However, these registers can be read and written in halt state by executing forced microinstructions (see [Forced microinstruction execution](#)). Immediate data microinstructions may be used to set register values. Some registers are not selectable for immediate data destination, so intermediary register(s) - notably P - may have to be used to carry the desired new value to the target register in two or more microinstructions. Usually the previous values of intermediary register(s) must be previously saved and restored after the whole operation.

Similar procedures apply for register reads: their contents must be dumped to SDM, where they can be read from the slave bus.

37.15.11.9 Microengine Flag Access

Microengine halt state allows reading the branch conditions flags through forced microinstructions or, more easily, through the NDEDI register NDEDI_ENGINEx_CFSR (see the NDEDI details). Flag conditions set by the user are seen by microengine for the next microinstruction execution. The flag set options are limited by the possibilities of forced microinstruction execution.

If the eTPU runs (not single-stepping) after exiting the halted state, the conditions modified during halt may remain only for the first microcycle after the halted state. After the first microcycle, branch conditions are altered only according to their regular update scheme.

37.15.11.10 Microengine Stall

Microengine can get into a stall state, attending a request from a debug interface signal assertion. The reason for a Stall request from NDEDI (or from any other debug support block) should be a temporary lack of resources, for instance queue full. During stall the microengine suspends execution, but all the other Engine logic continues operating: time

bases, angle logic, channel logic, input sampling and filters. Stall differs from Halt, not enabling any of the debug features that Halt enables (see [Microengine Halt State](#)). It also does not break an atomic microengine access, unlike halt.

The Microengine can be stalled when idle and from the moment TST ends, before executing the first thread microinstruction, until just before the last thread microinstruction is executed. Stall requests are ignored in any other occasions.

Microengines in a dual engine system can be independently stalled. If a forced end is issued when microengine is in stall coming from execution, the END is executed only when the microengine resumes execution from stall.

37.15.12 SCM Emulation

If SCM is implemented as ROM, an external RAM may be used to replace it, allowing code patching and software breakpoint setting for debugging purposes. SCM ROM replacement by Emulation RAM is MCU-dependent. The SCM may even be divided into a ROM part and a RAM part. In this case, both microengines can run code from both ROM and Emulation RAM. It is possible to make one Engine run code from RAM and the other from ROM, by using different Entry Tables. The SCM visibility conditions also apply to Emulation RAM.

All SCM implementations, either RAM, ROM or Emulation RAM, are external to the eTPU block. eTPU provides a signal to enable the switching between external SCM banks. The conditions for this switching are:

1. Both Engines stopped.
2. VIS bit = 0.

Note that these conditions also stop the clocks of the SCM interface and MISC logic.

37.15.13 Safety features

This section describes embedded safety features: MISC and memory error support.

The Multiple Input Signature Calculator - MISC - is an SCM test feature accessible through [eTPU Module Configuration Register \(eTPU_SCR_MCR\)](#) and [eTPU MISC Compare Register \(eTPU_SCR_MISCCMPR\)](#). MISC allows SCM test "on the fly", i.e., while eTPU is running, with no impact on eTPU functionality or performance.

Memory Error support features comprises SCM and SDM error detection, correction, report, and soft error.

37.15.13.1 SCM Test - Multiple Input Signature Calculator

The Multiple Input Signature Calculator (MISC) comprises special hardware that sequentially reads all SCM positions and calculates, in parallel, a 32-bit signature from a 32-input CRC signature calculator with the following polynomial:

$$1 + x^1 + x^2 + x^{22} + x^{31}$$

A complete description of the signature calculation procedure can be found in [MISC Algorithm](#).

Once started by the Host the MISC runs continuously, restarting after the completion of each cycle, when it sets [SCMMISC](#). The average time for a MISC calculation can be measured by checking SCMMISC state at regular intervals, incrementing a counter and clearing SCMMISC if it is set.

MISC accesses to the SCM array are executed if none of the engines is accessing the SCM, to avoid degradation of the microengine performance: it happens while no channel is being serviced. An ongoing MISC operation can be aborted by writing 0 to SCMMISEN.

The Host must load [eTPU MISC Compare Register \(eTPU_SCR_MISCCMPR\)](#) with the expected value to be found at the end of the MISC cycle, and then start the signature calculation by programming field SCMMISEN=1 in [eTPU Module Configuration Register \(eTPU_SCR_MCR\)](#). MISC zeroes the signature accumulator and starts reading SCM data and calculating the signature. After last SCM position is read, MISC compares the value in signature accumulator against the value in eTPU_SCR_MISCCMPR: if there is a mismatch MISC stops, a Global Exception is issued and [SCMMISF](#) assumes value 1. If no mismatch is found, MISC repeats the procedure automatically. When signature is being calculated, SCM address starts at the last SCM address and counts down to 0. The conditions for executing a MISC operation are (see also [Table 37-335](#)):

- Both microengines in idle state (no channel is being serviced) or stopped, in any combination (e.g., engine 1 idle with engine 2 stopped).
- [VIS](#) =0.
- [SCMMISEN](#) =1.

Note that MISC can run regardless of SCM implementation type (RAM or ROM).

If SCMMISEN=0 or VIS=1, the MISC logic stays at its initial state, with address counter pointing to the last SCM position and accumulator reset.

MISC can also be used to perform single-bit error fixes in the SCM. For more details, see [Error fixing](#).

37.15.13.2 Memory error support

Error Correction Code (ECC) support for SDM and SCM provides guaranteed single bit error correction and double bit error detection. Error detection and correction is performed on read accesses and on write accesses less than 32-bit wide to the SCM.

SDM ECC is calculated per byte, 5 bits wide each, totalling 20 ECC bits for each 32-bit word. That allows the ECC to be updated without a read-modify-write operation on writes of any size.

Error is not checked on bytes not selected on SDM read accesses, either by the Host, CDC or microengine. For instance, the most significant byte is not checked for errors on a microengine 24-bit read access.

Error detection also works for host accesses to the PSE area (see [Parameter Sign Extension area](#)). In this case, error is not detected in the most significant byte (bits 31:24). Also, because of sign extension, error is always detected on the bits 23:16, regardless if they are selected, whenever the most significant byte is read.

SCM ECC is 7 bits wide, comprising the whole 32-bit word. Read-modify-write is automatically made when 16-bit or 8-bit wide writes are made into the SCM in a host access. SCM errors are also detected on Entry Table accesses during TST (see [Time Slot Transition](#)) and on MISC accesses, which are considered microengine accesses with regards to interrupt generation and error reporting.

37.15.13.2.1 Error detection disabling

The error detection logic in SDM and SCM can be disabled by setting the [eTPU Memory Error Control Register \(eTPU_MESR_MECR\)](#) bits DEDD and CEDD, respectively. The disabled state turns ineffective any action due to error detection, including error correction. Note that error detection is always enabled out of reset.

37.15.13.2.2 Memory error flags

SDM single (correctable) and non-correctable bit errors are flagged in [eTPU Memory Error Status Register \(eTPU_MESR_MESR\)](#) fields DCERR and DNCERR, respectively.

SCM correctable and non-correctable errors are flagged in the eTPU_MESR_MESR fields CCERR and CNCERR, respectively.

Double bit errors are guaranteed to be flagged, while three or more simultaneous bit errors in an SDM byte or SCM word are not. An error flagged as non-correctable can be in two or more bits.

Overflow flags DCOVR and CCOVR indicate, for SDM and SCM respectively, when a correctable error occurs when DCERR, CCERR are already asserted. Similarly, overflow flags DNCOVR/CNCOVR assert when a non-correctable error occurs when DNCERR/CNCERR are asserted already.

37.15.13.2.3 Memory error reports

The address, syndrome and data of a non-correctable SDM access can be sampled into [eTPU Data Error Report Address Register \(eTPU_MESR_DERAR\)](#), [eTPU Data Error Report Syndrome Register \(eTPU_MESR_DERSR\)](#), and [eTPU Data Error Report Data Register \(eTPU_MESR_DERDR\)](#), respectively. eTPU_MESR_DERAR also captures the SDM read access originator, whether the host, CDC, or specific eTPU channel service.

Similarly, non-correctable SCM accesses address, syndrome and data are stored in [eTPU Code Error Report Address Register \(eTPU_MESR_CERAR\)](#), [eTPU Code Error Report Syndrome Register \(eTPU_MESR_CERSR\)](#), and [eTPU Code Error Report Data Register \(eTPU_MESR_CERDR\)](#), respectively. eTPU_MESR_CERAR also captures the SCM read access originator, whether the host, MISC, or specific eTPU channel service. The occurrence of a non-correctable error overwrites the report register values from a correctable error, either on SDM (DCERR=1) or SCM (CCERR=1).

The last address, syndrome and data of a correctable SDM access can also be sampled into eTPU_MESR_DERAR, eTPU_MESR_DERSR, and eTPU_MESR_DERDR, but only if [DNCERR](#) is 0. That prevents access address, syndrome and data from a correctable error to overwrite the same information from a non-correctable access.

Similarly, the last address, syndrome and data from a correctable SCM access are stored in eTPU_MESR_CERAR, eTPU_MESR_CERSR, and eTPU_MESR_CERDR only if [CNCERR](#) is 0.

For either SDM or SCM, a new correctable error overwrites the report registers' values from an earlier correctable error, but report values from a non-correctable error are not overwritten, as long as bit DNCERR/CNCERR is asserted, neither by a new correctable error nor by a non-correctable one.

The sampling of SDM and SCM error report registers can be disabled through the eTPU_MESR_MECR fields DRDIS and CRDIS, respectively. When DRDIS (CRDIS) is asserted, the update of the SDM (SCM) error report registers is disabled, but not of

eTPU_MESR_MESR fields DCERR, DNCERR (CCERR, CNCERR). DRDIS, CRDIS must be set before reading the respective report registers to assure their coherent reading, preventing them to be updated between reads.

37.15.13.2.4 Error detection and wait-states

Microengine SDM accesses do not take any wait-state due to ECC support. Host SDM read accesses take two wait-states minimum³⁹ in MCUs with SDM ECC support, regardless if an error was detected or not, if the SDM error detection is enabled (eTPU Memory Error Control Register (eTPU_MESR_MECCR) field DEDD=0). Host SDM read accesses take a minimum of one wait-state if the SDM error detection is disabled (eTPU_MESR_MECCR[DEDD]=1). Host SDM write accesses take one wait-state minimum, regardless of ECC support presence or enabling.

Host SCM read accesses have two wait-states³⁹, regardless if an error was detected or not. Writes from the host to the SCM take one wait-state on 32-bit accesses, and one additional wait-state (two wait-states total³⁹ on 16-bit and 8-bit wide accesses. Those figures are not affected by the SCM error detection disabling (eTPU_MESR_MECCR[CEDD]).

37.15.13.2.5 Memory error interrupts

For **microengine accesses**, the Global Exception interrupt asserts whenever a non-correctable error is detected, either on SCM or SDM (see [Interrupts and data transfer requests](#)). These errors are flagged by eTPU Module Configuration Register (eTPU_SCR_MCR) fields SCMERR and SDMERR, respectively. Single bit errors are corrected and activate the Memory Error Correction Interrupt if it is enabled (see [Interrupt types and sources](#)), but do not generate Global Exceptions.

For **host accesses** a bus error is issued whenever a non-correctable error is detected, either on SCM or SDM (but not through the CDC) and it also generates a Global Exception. Single bit errors are corrected (if the error correction feature is available) and do not cause bus transfer errors, but assert the Memory Error Correction Interrupt if it is enabled (see [Interrupt types and sources](#)).

37.15.13.2.6 Effect of errors on microcode execution

Correctable errors, either on SCM or SDM, do not affect microcode execution if detection/correction is enabled.

The executing thread is affected by a non-correctable SDM error as follows:

39. One wait-state more, compared with eTPU versions before eTPU2 and previous eTPU2 without ECC support

- If **DTEND** = 1: on SDM errors due to a read operation, if the instruction is not also an END, the following instruction automatically executes an END, and its operations may be executed, totally or partially. On SDM errors during the TST preload, the thread automatically ends on the first instruction, if the error occurs on a preload parameter selected by the entry point bit PP (see Entry Point Format section.) The first instruction may be executed totally or partially. Data read into P or DIOB may be corrupted in either case.
- If **DTEND** = 0: the TST or thread continues the execution flow normally. Data read into P or DIOB may be corrupted in either case.

The executing thread is affected by a non-correctable SCM error as follows:

- If **CTEND** = 1: on SCM errors, the corrupted instruction is not executed, instead replaced by a simple END. Due to instruction prefetching, an error may be detected and yet not affect the thread execution. However, an error detected in a prefetched instruction may cause error report and global or memory error interrupts, even if not executed. If a non-correctable error occurs on the entry table read during TST (see Time Slot Transition section), the first instruction in the thread is replaced with a simple END, a global exception is issued and the reports and flags are normally updated.
- If **CTEND** = 0: the execution flow continues normally, and the results depend on the particular instruction corruption.

Table 37-418. Error support summary if CEDD = DEDD = 0

Action		SCM		SDM	
		single bit error	double bit error	single bit error	double bit error
Memory Error Correction Interrupt		Asserts	Not asserted	Asserts	Not asserted
Global Exception		Not asserted	Asserts	Not asserted	Asserts
Bus Error (host reads)		Not issued	Issued	Not issued	Issued
Thread	CTEND = 0	Normal execution	Potential erroneous execution ¹	Normal execution	see DTEND
	CTEND = 1		Terminates before erroneous execution ²		
	DTEND = 0		see CTEND		Potential erroneous execution
	DTEND = 1				Terminates after erroneous read ³
Code Error Report Registers	CNCERR = 0	Updates eTPU_MESR_CERAR, eTPU_MESR_CERSR, eTPU_MESR_CERDR		Not related	
	CNCERR = 1	eTPU_MESR_CERAR, eTPU_MESR_CERSR, eTPU_MESR_CERDR not updated			

Table continues on the next page...

Table 37-418. Error support summary if CEDD = DEDD = 0 (continued)

Action		SCM		SDM	
		single bit error	double bit error	single bit error	double bit error
Data Error Report Registers	DNCERR = 0	Not related		Updates eTPU_MESR_DERAR, eTPU_MESR_DERSR, eTPU_MESR_DERDR	
	DNCERR = 1			eTPU_MESR_DERAR, eTPU_MESR_DERSR, eTPU_MESR_DERDR not updated	

1. The thread executes correctly if the error is detected on an instruction fetched but not executed.
2. The erroneous instruction is not executed at all, and replaced by a simple END.
3. The instruction following the erroneous read may be executed totally or partially.

37.15.13.2.7 Error Correction Code (ECC) and syndrome definition

SDM ECC is composed of 5 Hsiao code⁴⁰ bits for each data byte. SCM ECC is composed of 7 Hsiao code bits for each 32-bit word.

Table 37-419 shows the encoding of the SDM syndrome and global parity fields as they read on eTPU Data Error Report Syndrome Register (eTPU_MESR_DERSR). There is one syndrome and one global parity for each byte in an SDM word, numbered SYND0 to SYND3 from the least to the most significant.

Table 37-419. SDM syndrome definition

SYNDn (hex)	Result	SYNDn (hex)	Result
0E	error on data bit 8n+7	10	error on code bit 8n+4
15	error on data bit 8n+6	08	error on code bit 8n+3
07	error on data bit 8n+5	04	error on code bit 8n+2
1A	error on data bit 8n+4	02	error on code bit 8n+1
19	error on data bit 8n+3	01	error on code bit 8n
13	error on data bit 8n+2		
16	error on data bit 8n+1	00	no error
1C	error on data bit 8n	any other	non-correctable error

The following table shows the definition of the SCM syndrome field as it reads on eTPU Code Error Report Syndrome Register (eTPU_MESR_CERSR).

40. Hsiao, M.Y.; A Class of Optimal Minimum Odd-weight-column SEC-DED Codes; IBM Journal of Research and Development http://ieeexplore.ieee.org/xpl/freeabs_all.jsp?arnumber=5391627

Table 37-420. SCM syndrome definition

SYND (hex)	Result	SYND (hex)	Result	SYND (hex)	Result
40	error on code bit 6	58	error on data bit 25	46	error on data bit 11
20	error on code bit 5	0B	error on data bit 24	4A	error on data bit 10
10	error on code bit 4	54	error on data bit 23	52	error on data bit 9
08	error on code bit 3	15	error on data bit 22	62	error on data bit 8
04	error on code bit 2	16	error on data bit 21	13	error on data bit 7
02	error on code bit 1	34	error on data bit 20	29	error on data bit 6
01	error on code bit 0	25	error on data bit 19	31	error on data bit 5
		26	error on data bit 18	43	error on data bit 4
49	error on data bit 31	64	error on data bit 17	45	error on data bit 3
0D	error on data bit 30	2C	error on data bit 16	19	error on data bit 2
0E	error on data bit 29	1A	error on data bit 15	51	error on data bit 1
38	error on data bit 28	23	error on data bit 14	61	error on data bit 0
4C	error on data bit 27	2A	error on data bit 13	00	no error
1C	error on data bit 26	32	error on data bit 12	any other	non-correctable error

37.15.13.2.8 Error injection

Errors can be injected into the SDM read data path through [eTPU Data Error Injection Address Register \(eTPU_MESR_DEIAR\)](#), [eTPU Data Error Injection Data Pattern Register \(eTPU_MESR_DEIDPR\)](#) and [eTPU Data Error Injection Parity Pattern Register \(eTPU_MESR_DEIPPR\)](#). They define bit flips on accesses at the specific address specified in eTPU_MESR_DEIAR.

SDM error injection is enabled independently for host and microengine accesses by the [eTPU Module Configuration Register \(eTPU_SCR_MCR\)](#) fields HDEIE and MDEIE, respectively. The user must program eTPU_MESR_DEIAR, eTPU_MESR_DEIDPR, and eTPU_MESR_DEIPPR with HDEIE=MDEIE=0, and only then enable error injection by setting HDEIE=1 and/or MDEIE=1 to avoid races (see [eTPU Memory Error Status Register \(eTPU_MESR_MESR\)](#)).

Similarly, SCM errors can be injected into the SCM read data path through [eTPU Code Error Injection Address Register \(eTPU_MESR_CEIAR\)](#), [eTPU Code Error Injection Data Pattern Register \(eTPU_MESR_CEIDPR\)](#) and [eTPU Code Error Injection Parity Pattern Register \(eTPU_MESR_CEIPPR\)](#), enabled by the [eTPU Module Configuration Register \(eTPU_SCR_MCR\)](#) field CEIE. Errors are injected on host accesses to the address specified in eTPU_MESR_CEIAR when **VIS** =1, or on microengine and MISC accesses when **VIS**=0 .

37.15.13.2.9 Error fixing

Single bit **soft errors** can be fixed by rewriting the right value, as corrected by the ECC logic, into the same address of the erroneous data. eTPU provides means to perform such fixes safely, coherently and concurrently with the normal operation.

SDM fixes can be done using the CDC (see [Coherent Dual-parameter Controller \(CDC\)](#)), limited to the lower 8 Kbytes of SDM. The Host application simply has to program a transfer with both origin and destination positions pointing to the address of the data to be fixed. Because the PBBASE field has an address granularity of 8 bytes, a word adjacent to the one to be fixed must also be rewritten. The CDC transfer is atomic, therefore the fix is done coherently and does not interfere with the application code or data flows.

SCM fixes can be done by programming [eTPU Code Error Fix Register \(eTPU_MESR_CEFR\)](#) (see [eTPU Code Error Report Syndrome Register \(eTPU_MESR_CERSR\)](#)). The fixes are performed by the MISC logic, which must be enabled (see [SCM Test - Multiple Input Signature Calculator](#)). The modes available, selected by [CFIXM](#), are:

- **Automatic Fix Mode (CFIXM=11)**: the MISC logic automatically rewrites the contents of an address whenever it finds a correctable error; the CFIXM field keeps its value if a fix is made.

37.15.14 Performance monitoring features

37.15.14.1 Idle Counter

The Idle Counters [eTPU Idle Register for Engine 1 \(eTPU_ER_IDLER_ENG1\)](#) and [eTPU Idle Register for Engine 2 \(eTPU_ER_IDLER_ENG2\)](#) continuously count microcycles in which the microengine is not busy with channel service. They can be used to measure the microengine utilization by rating the count measured during a period of time to the number of microcycles contained in the period. The Idle Counters do not count microcycles when the engine is stopped, or is in TST or halt states.

37.15.14.2 Latency Monitor

eTPU implements Latency Monitors to measure the time taken to service a request. Each Latency Monitor provides:

- The measure of the greatest latency found during actual thread executions, accessed from [eTPU Latency Monitor Measure Register for Engine 1](#) (eTPU_LMR_LMnMR_ENG1) and [eTPU Latency Monitor Measure Register for Engine 2](#) (eTPU_LMR_LMnMR_ENG2).
- A comparison between the measured latency and a threshold, determined through [eTPU Latency Monitor Measure Register for Engine 2](#) (eTPU_LMR_LMnMR_ENG2) and [eTPU Latency Monitor Configuration Register for Engine 2](#) (eTPU_LMR_LMnCR_ENG2).

Each Latency Monitor works independently in one of the following modes, selected with the LMM field in [eTPU Latency Monitor Configuration Register for Engine 1](#) (eTPU_LMR_LMnCR_ENG1) and [eTPU Latency Monitor Configuration Register for Engine 2](#) (eTPU_LMR_LMnCR_ENG2) :

- **Disabled mode:** no latency measures or actions are taken.
- **Channel mode:** monitors the latency of an individual channel.
- **Priority mode:** monitors the combined latency of all channels in a given priority level - High, Medium or Low (see [Scheduler](#)).

Channel and Priority modes are detailed in the subsections below. The following applies to both modes:

- After each latency measurement, the value measured is compared with the LATENCY value in [eTPU Latency Monitor Measure Register for Engine 1](#) (eTPU_LMR_LMnMR_ENG1) and [eTPU Latency Monitor Measure Register for Engine 2](#) (eTPU_LMR_LMnMR_ENG2). If the new measured value is greater, it replaces the old value in [eTPU Latency Monitor Measure Register for Engine 1](#) (eTPU_LMR_LMnMR_ENG1) and [eTPU Latency Monitor Measure Register for Engine 2](#) (eTPU_LMR_LMnMR_ENG2).
- After each latency measurement, the value measured is compared with the MAXLAT value in [eTPU Latency Monitor Configuration Register for Engine 1](#) (eTPU_LMR_LMnCR_ENG1) and [eTPU Latency Monitor Configuration Register for Engine 2](#) (eTPU_LMR_LMnCR_ENG2), and if the new measured value is greater:

- A Global Exception is issued.
- The [eTPU Module Configuration Register \(eTPU_SCR_MCR\)](#) flag LTO1 (engine 1) or LTO2 (engine 2) is set, independently of the LATENCY value in [eTPU Latency Monitor Measure Register for Engine 1 \(eTPU_LMR_LMnMR_ENG1\)](#) and [eTPU Latency Monitor Measure Register for Engine 2 \(eTPU_LMR_LMnMR_ENG2\)](#).

NOTE

The LATENCY value is updated even if MAXLAT=0.

NOTE

MAXLAT=0 just inhibits the assertion of the LTO1/2 flag and Global Exception due to the respective Latency Monitor. Other Latency Monitor instances with non-zero MAXLAT value may assert LTO1/2 and Global Exception.

37.15.14.2.1 Channel mode

When the Latency Monitor operates in Channel mode:

- Only the channel selected by the MONCHAN field in [eTPU Latency Monitor Configuration Register for Engine 1 \(eTPU_LMR_LMnCR_ENG1\)](#) and [eTPU Latency Monitor Configuration Register for Engine 2 \(eTPU_LMR_LMnCR_ENG2\)](#) is monitored.
- Any combination of Host Service Requests (HSR), Channel Service Requests and Link Service Requests can be selected for monitoring, based on the SRSEL field in [eTPU Latency Monitor Configuration Register for Engine 1 \(eTPU_LMR_LMnCR_ENG1\)](#) and [eTPU Latency Monitor Configuration Register for Engine 2 \(eTPU_LMR_LMnCR_ENG2\)](#).
- The latency is counted in microcycles, from the moment the logic OR of all the service requests selected by SRSEL asserts to the moment this logic OR negates.
- The latency count is interrupted when the channel is disabled, either by programming the CPR field in [eTPU Channel n Configuration Register for Engine 1 \(eTPU_CCCR_CnCR_ENG1\)](#) and [eTPU Channel n Configuration Register for Engine 2 \(eTPU_CCCR_CnCR_ENG2\)](#) to 0, or by a watchdog timeout. Latency is counted until the channel is disabled and is not discarded; a new count can start from zero when the channel is re-enabled.

NOTE

If a selected service request type or a combination of them continues asserted during and after its service thread execution,

its latency counter will keep on running; this can happen, for instance, if either matches can service requests (e.g. channel modes `em_st/em_dt`), one match happens before the other is serviced, and only one of them is cleared. The Latency Monitor can, in cases like this, yield pessimistic measurements.

NOTE

The Latency Monitor in channel mode must be disabled before the channel it monitors is disabled.

37.15.14.2.2 Priority mode

When the Latency Monitor operates in Priority mode:

- Only the combination of channels assigned with the priority selected by the `SRSEL` field in [eTPU Latency Monitor Configuration Register for Engine 1](#) (`eTPU_LMR_LMnCR_ENG1`) and [eTPU Latency Monitor Configuration Register for Engine 2](#) (`eTPU_LMR_LMnCR_ENG2`) is monitored.
- The latency is measured in microcycles, starting from either of these two events, whichever happens first:
 - The logic OR of service requests of the selected priority transitions from 0 to 1 at a moment when all their grants are at state negated.
 - The grants of the selected priority are cleared by the scheduler at the end of the thread, and any of the service requests at the same priority is still active soon after the thread ends.
- The latency is measured until the grants of the selected priority are cleared by the scheduler, after the measurement started.

Note

A Latency Monitor in priority mode must be disabled before any of the channels in the selected priority is disabled, or any disabled channel is assigned with the selected priority.

37.15.14.3 Watchdog Monitor Operation

The watchdog mechanism can also be used to monitor, without interfering with the application execution, whether the execution time of any individual thread or a sequence of threads reaches a given limit. This can be achieved by asserting the WMO field in [eTPU Watchdog Timer Register for Engine 1 \(eTPU_ER_WDTR_ENG1\)](#) and [eTPU Watchdog Timer Register for Engine 2 \(eTPU_ER_WDTR_ENG2\)](#).

37.16 Initialization/application information

37.16.1 Configuration sequence

After initial power-on reset the eTPU remains in an idle state⁴¹, requiring initialization of several registers before any function can begin execution. Also, if the SCM is implemented in RAM, it should be initialized with the eTPU application code prior to configuring the eTPU. Configuration procedures are summarized as follows:

- If SCM is implemented as RAM, load the eTPU application code (see [SCM access](#)).
- Initialize SDM (optional if no ECC support available)
- Initialize the SCM MISC logic (optional, see [SCM Test - Multiple Input Signature Calculator](#)).
- Initialize Watchdog and Latency Monitors (optional).
- Initialize [eTPU Engine 1 Time Base Configuration Register \(eTPU_TBR_TBCR_ENG1\)](#) and [eTPU Engine 2 Time Base Configuration Register \(eTPU_TBR_TBCR_ENG2\)](#) to set up:
 - TCR1 and TCR2 prescalers and clock sources.
 - Select digital filtering mode.
 - TCRCLK signal filter control.
 - Angle mode operation (if necessary).
- Initialize [eTPU Engine 1 Configuration Register \(eTPU_SCR_ECR_ENG1\)](#) and [eTPU Engine 2 Configuration Register \(eTPU_SCR_ECR_ENG2\)](#) to set up:

41. Except when device debug request is asserted on power-on reset: in this case, the microengines wake-up in halt state.

- Entry table base.
- Filter prescaler clock control.
- Initialize eTPU Engine 1 STAC Configuration Register (eTPU_TBR_STAC_ENG1) and eTPU Engine 2 STAC Configuration Register (eTPU_TBR_STAC_ENG2) if you need to set up TCR1/2 resource Client/Server operation.
- Write to eTPU Channel n Configuration Register for Engine 1 (eTPU_CCCR_CnCR_ENG1) and eTPU Channel n Configuration Register for Engine 2 (eTPU_CCCR_CnCR_ENG2) to choose the Function to be performed by each channel, and its parameter base address.
- Write to eTPU Channel n Status Control Register for Engine 1 (eTPU_CCCR_CnSCR_ENG1) and eTPU Channel n Status Control Register for Engine 2 (eTPU_CCCR_CnSCR_ENG2) to choose among the possible variations within the function flow (FM bits).
- Write to SDM for parameter initialization of each configured channel.
- Write to eTPU Watchdog Timer Register for Engine 1 (eTPU_ER_WDTR_ENG1) and eTPU Watchdog Timer Register for Engine 2 (eTPU_ER_WDTR_ENG2) if you need to enable and set up the Watchdog(s) mode and timeout.
- Write to eTPU Channel n Host Service Request Register for Engine 1 (eTPU_CCCR_CnHSRR_ENG1) and eTPU Channel n Host Service Request Register for Engine 2 (eTPU_CCCR_CnHSRR_ENG2) to initialize the active channels.
- Write to eTPU Channel Interrupt Enable Register for Engine 1 (eTPU_GCR_CIER_ENG1) and eTPU Channel Interrupt Enable Register for Engine 2 (eTPU_GCR_CIER_ENG2) if interrupts are to be enabled from the appropriate channels. Likewise for Data Transfer Requests (eTPU Channel Data Transfer Request Enable Register for Engine 1 (eTPU_GCR_CDTRER_ENG1) and eTPU Channel Data Transfer Request Enable Register for Engine 2 (eTPU_GCR_CDTRER_ENG2)). This can also be done through eTPU Channel n Configuration Register for Engine 1 (eTPU_CCCR_CnCR_ENG1) and eTPU Channel n Configuration Register for Engine 2 (eTPU_CCCR_CnCR_ENG2).
- Write to eTPU Channel n Configuration Register for Engine 1 (eTPU_CCCR_CnCR_ENG1) and eTPU Channel n Configuration Register for Engine 2 (eTPU_CCCR_CnCR_ENG2) to enable each channel by assigning it a high, middle, or low priority (CPR field).⁴¹

- Monitor [eTPU Channel n Host Service Request Register for Engine 1 \(eTPU_CCCR_CnHSRR_ENG1\)](#) and [eTPU Channel n Host Service Request Register for Engine 2 \(eTPU_CCCR_CnHSRR_ENG2\)](#) for completion of initialization.
- Write [GTBE](#) =1 to start TCR1/TCR2 time base counting at same time in both Engines (may be done before or never, depending on the particular application and use of Red Line bus).

See [Initialization Code Example](#).

37.16.2 Reset options

37.16.2.1 Hardware reset

Hardware reset is achieved by assertion of device synchronous reset. Both Engines and common logic is reset, and even the System Configuration and Global Channel registers assume their reset values.

Note

All eTPU input clocks must pulse during reset so that both engines are reset, even if they are in Module Disable or Stop mode.

37.16.2.2 Software reset

eTPU has no software reset. To abort infinite microcode loops, the Force END mechanism must be used (see field FEND in [eTPU Engine 1 Configuration Register \(eTPU_SCR_ECR_ENG1\)](#) and [eTPU Engine 2 Configuration Register \(eTPU_SCR_ECR_ENG2\)](#)).

37.16.3 Multiple parameter coherency methods

Follows a description of two methods for coherent transfer of multiple parameters between Host and eTPU. Both methods involve the use of two parameter areas: the Transfer Parameter Area (hereafter called TPA), which is the SDM area directly accessed by the Host for reads and writes, and the Permanent Parameter Area (hereafter called

PPA), which are the SDM positions where channel parameters are normally accessed by the Function microcode. Note that parameters in either TPA or PPA do not have to be in sequential addresses. TPAs and PPAs allocation are completely defined by the application, and there may be any number of them, independently of the channels.

The methods described here are not the only solutions for the coherent transfer problem, and both can coexist in eTPU and even used in combination. Also note that for transfers of a pair of parameters, the Coherent Dual-parameter Controller is faster and have less impact on both eTPU and Host performance. That said, the methods are:

- **Transfer Service:** a microengine thread transfers, upon Host Service Request, data from/to a TPA to/from a PPA. Coherency is guaranteed by the fact that a thread is atomic with respect to other threads in the same Engine, and so are its transfers. If parameters in PPA are shared by both Engines, hardware semaphores have to be used to access them.
- **Mailbox:** for Host to eTPU transfers, the microcode checks a flag, set by the host, indicating the existence of new parameter data in the TPA. It can, then, either access TPA data directly or copy it to the PPA. For eTPU to Host transfers, when microcode changes PPA, it copies them to the TPA and flags updated TPA data to Host, possibly using an Interrupt or a Data Transfer Request. The Mailbox flag is reset when data is copied: by the eTPU microcode, when it transfers TPA to PPA (possibly followed by an Interrupt); by the Host, when it reads data from the TPA. This indicates that TPA is free for another transfer.

Transfer Service has the advantage of separating the task of data transfer from the functional service thread that accesses the parameters, with less impact to the latter. Compared to the Mailbox method, however, it has bigger average latency, because the Transfer Service thread has to contend for a time slot to execute. This latency can be minimized if Transfer Service thread is assigned to a separate channel with higher priority, but even so it does not guarantee that PPA is updated before the next execution of the functional thread that uses it.

Mailbox method, on the other hand, makes the functional thread check for the existence of new data (Host to eTPU). It does not have to be responsible for the transfer, though: it may access the TPA directly, and a Transfer Service can then be used to copy data from TPA to PPA.

37.16.4 Programming Hints and Caveats

37.16.4.1 Atomic dual access after a call, return

A dual, back-to-back parameter access is not atomic after a call, a jump, or a return if they occurred in parallel with an odd SDM access. It is safer to make a pair of parameter accesses that must be coherent begin at the second instruction after a call/jump/return.

37.16.4.2 Resource polling

The use of polling while waiting for a condition or a resource (except semaphore lock) should be avoided in order not to hang the microengine in long loops. This general programming guideline is greatly enforced in eTPU, as a thread cannot be preempted for any reason. Safer polling, albeit with long and indeterministic latency, can be obtained if a thread issues a channel link to the serviced channel and terminates. The microengine is then free to other tasks, and the next poll happens at the next time the channel is serviced. This mechanism can be combined with finite (timed out) loops for better latency.

37.16.4.3 Changing channel function, parameter base, or entry table scheme

Channel function, parameter base address and entry table scheme are determined by the CFS, CPBA, and ETCS fields in [eTPU Channel n Configuration Register for Engine 1 \(eTPU_CCCR_CnCR_ENG1\)](#) and [eTPU Channel n Configuration Register for Engine 2 \(eTPU_CCCR_CnCR_ENG2\)](#). They cannot be changed when the channel is enabled. If the channel is disabled first, one may still have service requests from the previous function, so before the channel is enabled again one must be sure that:

- the first thread executed in the new function is the initialization one.
- the initialization thread of the new function clears any previously pending service request.

Here is a safe procedure for function changing:

1. Disable the channel (in [eTPU Channel n Configuration Register for Engine 1 \(eTPU_CCCR_CnCR_ENG1\)](#) and [eTPU Channel n Configuration Register for Engine 2 \(eTPU_CCCR_CnCR_ENG2\)](#), write CPR=00).
2. Change the function configuration (fields CFS and/or CPBA and/or ETCS in [eTPU Channel n Configuration Register for Engine 1 \(eTPU_CCCR_CnCR_ENG1\)](#) and [eTPU Channel n Configuration Register for Engine 2 \(eTPU_CCCR_CnCR_ENG2\)](#)).

3. Request the initialization thread, writing [eTPU Channel n Host Service Request Register for Engine 1 \(eTPU_CCCR_CnHSRR_ENG1\)](#) and [eTPU Channel n Host Service Request Register for Engine 2 \(eTPU_CCCR_CnHSRR_ENG2\)](#) with the initialization HSR (channel still disabled).
4. enable the channel (in [eTPU Channel n Configuration Register for Engine 1 \(eTPU_CCCR_CnCR_ENG1\)](#) and [eTPU Channel n Configuration Register for Engine 2 \(eTPU_CCCR_CnCR_ENG2\)](#) write $CPR > 0$); the initialization HSR is serviced before any other formerly pending service requests, clearing them.

37.16.4.4 Checking and Clearing Interrupts of a Stopped Engine

An Engine may be stopped with interrupts (or DMA requests) pending. This includes the case when the Engine's MDIS bit is set and a thread is still running: the thread will complete execution, possibly issuing an interrupt or DMA request before the engine stops, setting the STF bit.

As soon as the engine stops the channel registers become inaccessible, issuing bus errors when accessed. Interrupts and DMA requests can still be checked and cleared through the Global Channel Registers, though. DMA requests can also be cleared by the hardware handshaking with the DMA controller when the engine is stopped.

37.16.5 Estimating Worst Case Latency

Reliable systems are designed to work under worst-case conditions. This section explains how to estimate worst-case latency (WCL) for any eTPU function in any system. The appendix covers the following topics:

- Introduction to Worst-Case Latency
- Using Worst-Case Latency Estimates to Evaluate Performance
- Priority Scheme Details used in WCL Analyses
- First-Pass WCL Analysis
- Second-Pass WCL Analysis

The first-pass WCL analysis is based on a deterministic, generalized formula that is easy to apply. Because of the generalizations in the formula, the first analysis result is almost always much worse than the real worst case. If the desired system performance is within the limits of this first analysis, then no further analysis is required; the system is well

within the performance limits of the eTPU. If the desired system performance exceeds that indicated by the first analysis, the second-pass WCL analysis should be applied. The second-pass analysis is not a generalized formula, but rather uses specific system details for a realistic worst-case estimation.

37.16.5.1 Introduction to Worst-Case Latency

Note

The latency calculation and examples refer to old TPU functions such as PWM, DIO etc. These functions use single action channels which have single transition and single match functionality. They are not optimized for the eTPU hardware enhancement which support various double action modes. These examples are for reference only. New eTPU functions which are optimized for the new hardware will impose different latency calculations.

Worst-case latency for a channel is the longest amount of time that can elapse between the execution of any two function threads on that channel. For example, if in a particular system, channel 5 is running PWM, the worst-case latency for channel 5 is the longest possible time between the execution of two PWM threads. The worst case time includes the time the execution unit takes to execute threads for other active channels, and other delays described later in this section. Refer to the following figure.

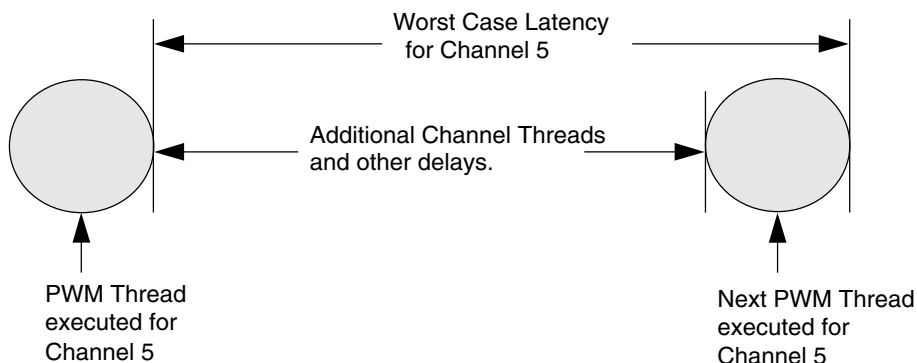


Figure 37-335. Worst-Case Latency for PWM

Worst-case latency for a channel depends both on the function running on that channel and on the activity on other channels. Since the 32 eTPU channels must all share the same execution unit, execution speed of a particular function varies with each system. The PWM thread response is faster if there are no other active channels than if other

channels are also active. In addition, changing the priority scheme and channel number assignments can change performance for a function even if the same set of functions are still active.

Each function is divided into threads, as shown in the figure below (see also [Functions and threads](#)). The eTPU Microengine executes one thread of a function at a time. For example, the Microengine might execute thread 1 of PWM, then thread 3 of DIO, then thread 2 of PWM, then thread 2 of SM, and so on. The amount of time the eTPU Microengine grants a function to execute a thread varies with the number of microcode instructions in the thread.

Since there is only one eTPU Microengine (in each eTPU Engine), the eTPU cannot actually execute the software for multiple functions simultaneously. However, the hardware for each of the channels is independent. This means that, for example, all 32 channel signals can change thread at the same moment, provided that the function software sets up the channel hardware to do so beforehand.

With Host CPU code, the system designer assigns functions to channels and initializes the functions. After initialization, functions typically run without Host intervention, except for eTPU channel interrupts to the Host to give or receive information. Most functions can run continuously with periodic servicing from the eTPU Microengine. As required, the channels request service from the eTPU Microengine, and the eTPU Scheduler determines the order in which the channels are serviced. Worst-case latency for a channel can be derived from the details of the priority scheme that the scheduler uses (see [Scheduler](#)).

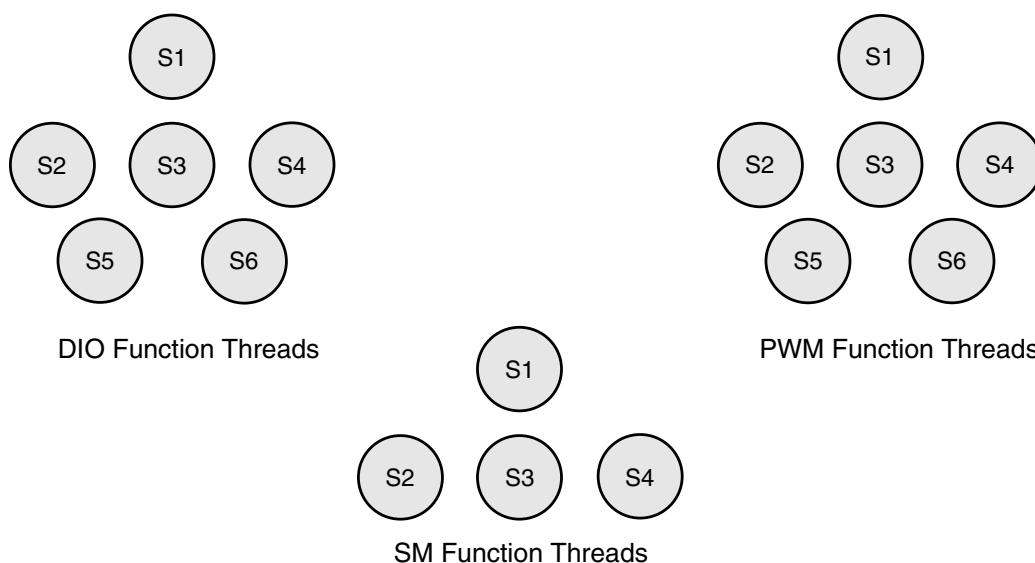


Figure 37-336. Function Threads

37.16.5.2 Using Worst-Case Latency estimates to evaluate performance

Once WCL is found for a channel, the user must determine how to use this number to analyze performance. To analyze the performance of a channel running the PWM function, for example, some information about what happens in each thread is necessary.

The following example refers to old TPU PWM function, which is not optimized to the eTPU enhanced hardware. For PWM, thread 1 is the initialization thread, and threads 2 and 3 are used during normal function execution. (PWM threads 4, 5, and 6 are for special modes and will be assumed to be unused on channel 5). Thread 2 writes a time into the channel 5 match register and performs other operations that will cause the channel 5 signal to go from low to high at the time indicated in the match register (match time). At match time, the signal goes high and channel 5 requests service from the eTPU Microengine to execute thread 3. Thread 3 writes a time into the channel 5 match register and performs other operations that will cause the channel 5 signal to go from high to low at match time. At match time, the signal goes low and channel 5 requests service from the eTPU Microengine to execute thread 2. A PWM wave is kept running on the system by the eTPU executing thread 2, then thread 3, then thread 2, then thread 3, and so on.

Since the definition of worse-case latency assumes a fully loaded running system, initialization threads are not part of worst-case calculations. For the channel 5 example, the two PWM threads in [Figure 37-335](#) are thus the two normal running threads, threads 2 and 3.

[Figure 37-335](#) does not define which thread is thread 2 and which is thread 3. Since the worst-case latency derived from the first-pass analysis is the worst case between any 2 threads (not counting initialization threads), it is safe to say that the worst-case latency shown in [Figure 37-336](#) represents both the worst-case high time and the worst-case low time.

Notice in [Figure 37-335](#) that worst-case latency is drawn from the end of the execution of the first PWM thread to the end of the execution of the next PWM thread. It is drawn from end to end because the microcode instructions that make up the threads control the channel hardware. To make sure that all the microcode instructions needed to change the pin thread have been executed, it is necessary to include the execution time of the second thread.

Thread information for each function is found in the programming notes for individual TPU functions. See NXP Programming Note TPUPN00/D, Using the TPU Function Library and TPU Emulation Mode, for a list of available programming notes. Similar documentation will be provided for the eTPU new functions.

37.16.5.3 Priority Scheme Details Used in WCL Analysis

The user assigns functions to channel numbers and gives each active channel a priority level of high, middle, or low. The Scheduler uses the channel number and channel priority level to determine the order in which to grant service.

The scheduler allocates time slots to specific priority levels of high, middle, or low. One function thread is executed in each time slot. The length of a time slot varies according to the length of the executing thread. When fully loaded, the scheduler always assigns time slots in a seven-slot sequence (see the following table). After a seven-slot sequence is completed, another seven-slot sequence begins (see [Figure 37-338](#)). Note that in eTPU, when no service request exists, the scheduler goes to thread 1, but WCL calculation considers full load.

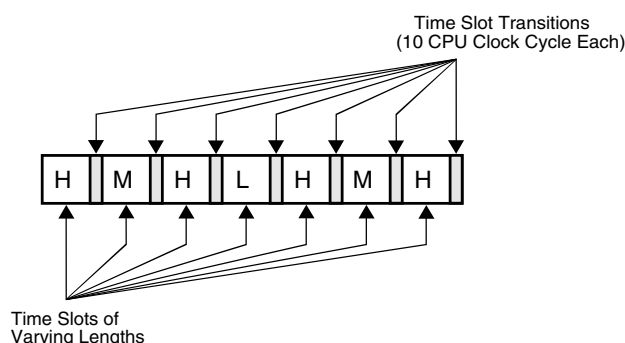


Figure 37-337. Time-Slot Sequence

This sequence scheme gives higher-priority channels more service time than lower-priority channels. High-priority channels are allocated four of seven time slots, middle-priority channels are allocated two of seven time slots, and low-priority channels are allocated one of seven time slots.

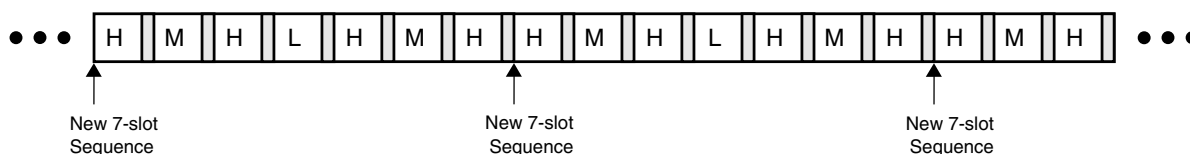


Figure 37-338. Multiple Time-Slot Sequences

37.16.5.3.1 Priority Passing

If no channel of the priority level assigned to the time slot is requesting service, the eTPU scheduler can pass priority to other levels. If no high-level channel is requesting service during a high level time slot, a middle-level channel is granted service; or, if no middle level-channel is requesting service, a low-level channel is granted service. If no middle-level channel is requesting service during a middle-level time slot, a high-level channel is granted service; or, if no high-level channel is requesting service, a low-level channel is

granted service. If no low-level channel is requesting service during a low-level time slot, a high-level channel is granted service; or, if no high-level channel is requesting service, a middle-level channel is granted service. If no channel is requesting service, the time slot sequence is reset to state 1 and the scheduler idles until a request is received.

Priority passing is implemented in hardware and does not contribute to worst-case latency.

37.16.5.3.2 Time-slot transition

After each time slot, the eTPU must prepare for the next time slot. This preparation time between each time slot is called a time-slot transition. See [Time Slot Transition](#). Time-slot transitions can take from six up to ten eTPU clocks.

37.16.5.3.3 Channel Number Priority

If more than one channel of a priority level is requesting service, the lowest numbered channel is granted service first. For example, if channels 0, 5, and 9 are all high-level channels requesting service during a high time slot, channel 0 is granted service first. Continuing this example, if channel 0 requests service again immediately after being serviced, it is not serviced again until channels 5 and 9 are serviced. This scheme is implemented so that continuously-requesting low numbered channels do not take all the time on the eTPU execution unit and leave no time for other channels.

The scheduler uses registers to keep track of which channels have been serviced and which require servicing. Each channel has two register bit: a service request register (SRR) and a service grant register (SGR). The SRR is set when a channel requests service. After the channel has been granted service, the SGR is set and the SRR is cleared.

SGRs are not cleared individually by channel, but rather as priority level groups. The clearing of a group of SGRs begins a new cycle for that priority level. An SGR group is cleared on the condition that a channel of that priority level has just been serviced, and no other channel of that priority level is requesting service (has a set SRR) and has not been granted service (has a clear SGR).

For example, if a middle-priority channel has just been serviced (either in a middle-priority time slot or a high or low-priority time slot gained by priority passing), the SRRs and SGRs of all middle-priority channels are compared. If there is no middle-priority channel with its SRR set and SGR cleared, the scheduler clears all middle-level SGRs. If there is a middle-level channel with its SRR set and SGR cleared, the scheduler does not clear the SGR group, and the requesting middle-level channel is serviced on the next middle-level time slot (or possibly sooner by priority passing).

37.16.5.3.4 SDM Collision Rate

Most function threads read or write to the eTPU SDM at least once. Because both the eTPU Microengine and Host can access the SDM but not at the same time, the Microengine may suspend execution during the SDM access while waiting for the Host to finish accessing the SDM. At other times the Host may wait for the Microengine. Wait states can take up to two eTPU clocks, when the Host accesses the SDM directly, without using CDC. Microengine(s) wait-states must be added into the worst-case latency calculation. The system designer should estimate the percentage of SDM accesses in the system that will result in Microengine wait-states. This percentage is called the RAM collision rate (RCR). In each collision with direct Host accesses to the SDM the Microengine(s) wait for two eTPU clocks.

In eTPU the Coherent Dual-parameter Controller (CDC) may also access the SDM for atomic transfers of two parameters. eTPU Microengine may wait on this operation (if it is in service time) until the transfer is complete. CDC always transfers two parameters, making four consecutive accesses (read, write, read, write) of one eTPU clock each. The system designer should estimate the percentage of SDM accesses in the system that will result in a Microengine wait due to coherent transfer, and multiply it with the average number of eTPU clocks the Microengine waits for each transfer. This percentage is called Coherent Parameter Collision Rate (CPCR).

In addition, Microengine to Microengine multiple parameter coherent communication, using the hardware semaphores, may hold one Microengine which waits to lock the semaphore while the other Microengine is holding it. This waiting is due to a software loop, not hardware wait-states. Note that single parameter access of one Microengine does not affect the timing of the other Microengine due to SDM time interlace. This implies that single parameter Microengine to Microengine communication does not affect the performance. The Microengine which waits for the semaphore will loop until it is freed by the other Microengine. This time depends on the eTPU application. The system designer should estimate the percentage of Microengine to Microengine coherent parameter communication that will result in eTPU semaphore loops, and multiply it with the average number of eTPU clocks the Microengine loops for each such transfer. This percentage is called CCR (Communication Collision Rate).

A 100% collision rate for a system is the theoretical worst case. In many systems, however, the RCR, CPCR and CCR would be very low, sometimes even near 0%. This is because the eTPU is an independent processor capable of servicing most function needs, so that the Host rarely needs to access the eTPU SDM. Also coherent Microengine to Microengine communication of more than one parameter may be rare. To find a realistic

RCR, CPCR the system designer should evaluate the Host code and find the percentage of time it accesses the eTPU SDM with or without using the CDC. This percentage gives a good RCR and CPCR. The eTPU application provides a good estimation of CCR.

Note

The programming practice of polling a flag in the eTPU SDM causes a very high RCR and should be avoided in high-performance systems.

After the collision rate for a system is found, it can be applied to the WCL calculations for each channel. The system designer can use the collision percentage and the number of SDM accesses (with and without semaphores) to estimate the eTPU loop time for a function. Note that in old TPU functions CPCR and CCR are both zero.

The estimation of eTPU wait time is as follows:

Variables:

N1 = Number of simple RAM accesses in the longest thread

RCRWait = Maximal eTPU clocks wait time for simple RAM collision = 2

CPCRWait = Average eTPU clocks for Coherent Parameter Transfer (using CDC).

N2 = Number of eTPU-eTPU semaphore RAM accesses in the longest thread

CCRWait = Average eTPU clocks for Microengine-Microengine communication transfer.

Estimated Wait Time:

Function eTPU maximal wait time =

$N1 * (RCR * RCRWait + CPCR * CPCRWait) + N2 * CCR * CCRWait$

37.16.5.4 First-Pass Worst-Case Latency Analysis

Following is the first-pass calculation of worst-case latency for a channel. Remember that this analysis uses generalizations that usually produce a result much worse than the real worst case. If the worst-case result from the first analysis is too long for the desired performance, use the second analysis for a more realistic worst-case analysis.

37.16.5.4.1 Worst-case assumptions and formula

To estimate worst-case latency for a channel, assume this worst-case condition: the channel has just been serviced in a time slot of its priority level, and all other channels in the system are continuously requesting service and have cleared SGRs. The worst-case latency is the time from the end of the channel's service until the end of the channel's next service. See the following figure.

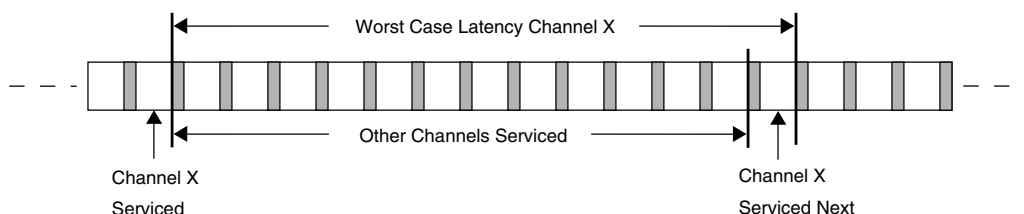


Figure 37-339. First-pass Worst-Case Latency

To estimate worst-case latency:

- Find the worst-case service time for each active channel.
- Using the H-M-H-L-H-M-H time-slot sequence, map the channels that are granted for each time slot.
- Add time for six-clock time-slot transitions.

37.16.5.4.1.1 Finding the Worst-Case Service Time for Each Active Channel

A table for eTPU functions should list the longest threads (not counting initialization threads) for the functions, and the number of eTPU SDM accesses in the longest thread (semaphored and non semaphored). These figures will be used for estimating Microengine wait time. The table below is an example for old TPU functions in which there are only simple SDM accesses. It does not take into consideration the CDC operation and Microengine to Microengine communication.

The worst-case service time for each channel is: (CPCR=CCR=0)

Longest thread + ((number of RAM accesses in longest thread+1) * RCR * 2 clocks).

Note that the formula adds 1 RAM accesses for the parameter preload that occurs during TST. There are actually three accesses during TST, but only the first one can receive wait-states.

Table 37-421. Longest Threads and RAM Accesses for old TPU Functions

Function	Longest Thread	RAM Accesses
DIO	10	4

Table continues on the next page...

Table 37-421. Longest Threads and RAM Accesses for old TPU Functions (continued)

Function	Longest Thread	RAM Accesses
ITC	40 (no linking) 42 (linking)	7
OC	40	7
PWM	24	4
SPWM	14 18 20 (no linking) 22 (linking)	4
Mode 0		
Mode1		
Mode 2		
PMA	94	8
PMM	94	8
PSP	76 50	6 3
Angle-Angle Mode		
Angle-Time Mode		
SM ¹	160	21
PPWA	44 50 ² 44 50	9 10 9 10
Mode 0		
Mode 1		
Mode 2		
Mode 3		

- Assumes one master and one slave. For each additional slave
 - Tab Add 32 clocks and 2 RAM accesses, and
 - Tab Add (STEP_RATE_CNT * two clocks)
- With one channel linked. Add two clocks for each additional channel linked.

37.16.5.4.1.2 Mapping the Channels for Each Time Slot

To determine when a channel will be serviced again, it is necessary to determine which other channels will be serviced first. Do this by assuming all channels are continuously requesting service and mapping the channels into the time-slot sequence.

37.16.5.4.1.3 Adding Time for Time-Slot Transitions

Add six eTPU clocks for time-slot transitions which occur after each time slot.

37.16.5.4.2 First-pass analysis Worst-Case Latency examples

The examples in this section assume the system configuration shown in the following table.

Table 37-422. System configuration example

Channel	Priority	Function ^{1, 2}
0	High	PWM (driving a DC motor)
1	Middle	PPWA (Mode 0, measuring the DC motor speed)
2	Low	DIO (Input)

1. 9% RAM Collision Rate (RCR)

2. CPU clock rate = 40 MHz, or 25 ns per clock period

37.16.5.4.2.1 Finding the WCL for PWM on Channel 0

The following shows how to find the WCL for PWM on channel 0.

1. Find the worst-case service time for each active channel.

a. Longest thread of PWM is 24 CPU clocks with four RAM accesses.

$24 + ((4 \text{ RAM accesses} + 1) * 0.09 * 2 \text{ CPU clock waits}) = 24.9 \text{ CPU clocks}$,
rounded up to 25 CPU clocks (since there are no partial clock periods)

Channel 0 worst-case service time = 25 CPU clocks.

b. Longest thread of PPWA in mode 0 is 44 CPU clocks with nine RAM accesses.

$44 + ((9 \text{ RAM accesses} + 1) * 0.09 * 2 \text{ CPU clock waits}) = 45.8 \text{ CPU clocks}$,
rounded up to 46 CPU clocks

Channel 1 worst-case service time = 46 CPU clocks.

c. Longest thread of DIO is ten CPU clocks with four RAM accesses.

$10 + ((4 \text{ RAM accesses} + 1) * 0.09 * 2 \text{ CPU clock waits}) = 10.9 \text{ CPU clocks}$,
rounded up to 11 CPU clocks

Channel 2 worst-case service time = 11 CPU clocks.

2. Assume channel 0 has just been serviced and that channels 1 and 2 are continuously requesting service. Using the H-M-H-L-H-M-H time-slot sequence, map the channels that are granted for each time slot. See the following figure.

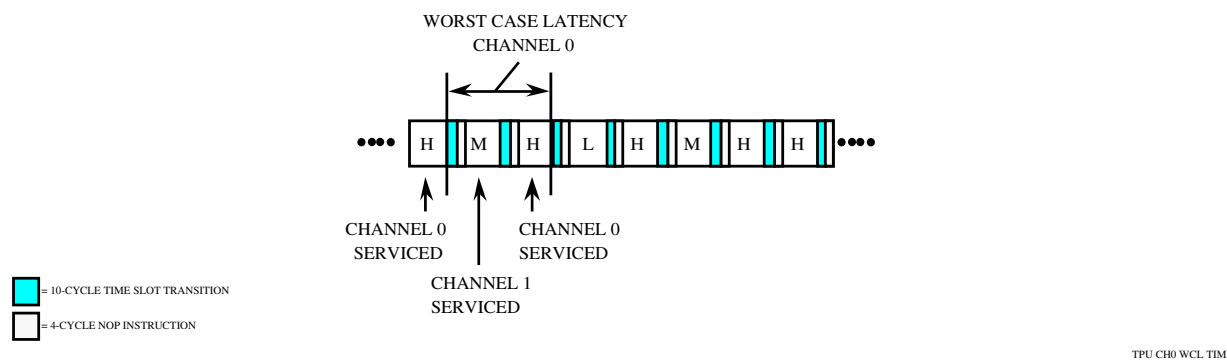


Figure 37-340. Next Servicing for Channel 0

Channel 1 will be serviced in the middle-priority time slot before channel 0 is serviced again.

3. Add time for the six-clock CPU time-slot transitions. See the above figure and the following table.

A four-clock NOP occurs after each channel is serviced since there is one channel in each priority level, i.e., a new cycle for a priority level is started after each channel is serviced. Time-slot transitions occur after each time slot.

Table 37-423. Worst-Case Latency for Channel 0

Channel 0 worst-case service time	25 clocks
Channel 1 worst-case service time	46 clocks
Two 6-clock time-slot transitions	12 clocks
Total clocks	83 clocks

$83 \text{ clocks} * 25 \text{ ns/clock} = 2075 \text{ ns}$

Conclusion: in this system configuration PWM can run with a minimum high time or low time of 2075 ns.

Note that in double match eTPU system the PWM can be serviced once in each period, and there is no latency for minimum high time. The latency in eTPU PWM function will represent the minimum PWM period.

37.16.5.4.2.2 Finding the WCL for PPWA on Channel 1

The following shows how to find the WCL for PPWA on channel 1.

1. Find the worst-case service time for each active channel. See step 1 of previous example.
2. Assume channel 1 has just been serviced and that channels 0 and 2 are continuously requesting service. Using the H-M-H-L-H-M-H time-slot sequence, map the channels that are granted for each time slot. See the following figure.

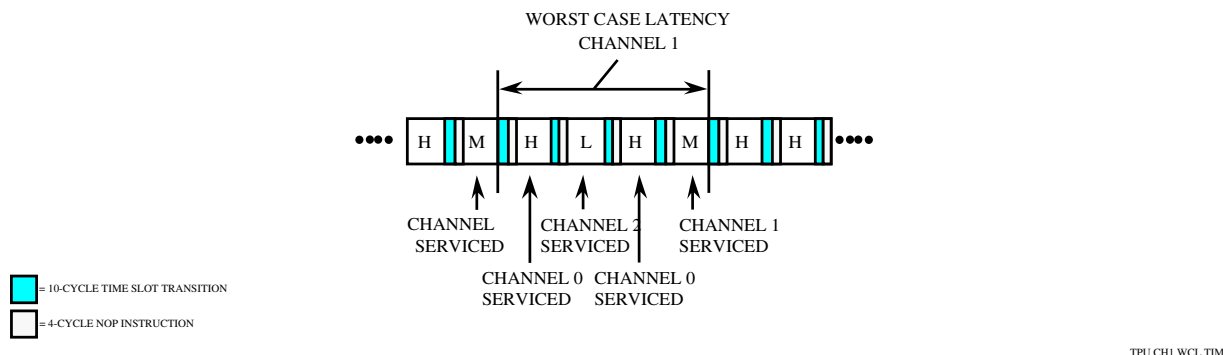


Figure 37-341. Next Servicing for Channel 1

Channel 0 will be serviced twice and channel 2 once before channel 1 is serviced again.

3. Add time for the six-clock CPU time-slot transitions. See the above figure and the following table.

Table 37-424. Worst Case Latency for Channel 1

Two Channel 0 worst-case service times	50 clocks
Channel 1 worst-case service time	46 clocks
Channel 2 worst-case service time	11 clocks
Four 6-clock time-slot transitions	24 clocks
Total clocks	131 clocks

$131 \text{ clocks} * 25 \text{ ns/clock} = 3275 \text{ ns}$

Conclusion: in this system configuration PPWA can measure a period or pulse of minimum 3275 ns.

Note that PPWA function optimized for eTPU hardware can use double transition mode to measure very narrow pulses with one service after the second transition, and latency will affect only the minimum gap between two input pulses. Also the function threads would have more efficient coding.

37.16.5.4.2.3 Finding the WCL for DIO on Channel 2

The following shows how to find the WCL for DIO on channel 2.

1. Find the worst-case service time for each active channel. See step 1 of previous examples.
2. Assume channel 2 has just been serviced and that channels 0 and 1 are continuously requesting service. Using the H-M-H-L-H-M-H-H-M-H-L-H time-slot sequence, map the channels that are granted for each time slot. See the following figure.

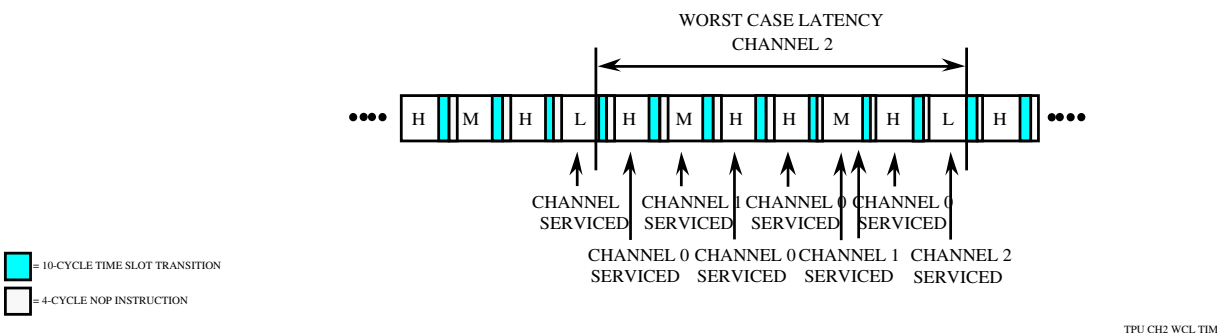


Figure 37-342. Next Servicing for Channel 2

Channel 0 will be serviced four times and channel 1 twice before channel 2 is serviced again.

3. Add time for the ten-clock CPU time-slot transitions and the four-clock NOPs. See the above figure and the following figure.

Table 37-425. Worst Case Latency for Channel 2

Four Channel 0 worst-case service times	100 clocks
Two Channel 1 worst-case service time	92 clocks
Channel 2 worst-case service time	11 clocks
Seven 6-clock time-slot transitions	42 clocks
Total clocks	245 clocks

$245 \text{ clocks} * 25 \text{ ns/clock} = 6125 \text{ ns}$

Conclusion: in this system configuration DIO can keep track of the input level at a minimum of every 6125 ns.

Note that DIO function optimized for eTPU hardware can use double transition mode to measure two pin transitions at a time and reduce the service time, improving the overall system performance and latency.

37.16.5.5 Second-Pass Worst-Case Latency Analysis

Following is an example of a second-pass analysis for calculating worst-case latency for a channel. The second-pass analysis is useful for higher-performance systems, since it gives a more realistic worst-case latency result than first-pass analysis.

This example uses a relatively simple system in order to illustrate the basic principles of second-pass analysis. For a more complex example of second-pass analysis, refer to Multiphase Motor Commutation TPU Function (COMM)(TPUPN09/ D).

37.16.5.5.1 Second-Pass Analysis Guidelines

Rather than use a fixed formula, a second-pass analysis relies on the application of the following guidelines.

1. The first-pass analysis makes the assumption that all channels in the system are continually requesting service. For many systems this is an unrealistic assumption. For example, if TCR1 is counting at a rate of 2 MHz (500 ns per count) and a channel is running the DIO function with a match rate of 20,000 TCR1 counts, the DIO will request service every 10 ms ($20,000 * 500 \text{ ns} = 10,000,000 \text{ ns}$ or 10 ms). It is therefore unrealistic to assume that the channel running this DIO function is continuously requesting service. Figure out a realistic service request rate for each channel. Time slots can then be mapped to each channel at the real rate of request.
2. If a function is active during system initialization but not during the high-speed running mode of the system, then that system does not need to be included in the high-speed worst-case latency calculations.
3. Use a realistic SDM collision rate.
4. Be careful when assigning functions priority levels and channel numbers. Decide which function or functions will be most difficult to make perform at the desired level. Assign those channels high priority and low channel numbers. Try different priority and channel assignments to see how it affects the system.
5. The seven-slot sequence of || H | M | H | L | H | M | H || is asymmetrical when put back-to-back with other seven-slot sequences. Note that in the following sequence there are two high-priority slots next to each other:

|| H | M | H | L | H | M | H ||| H | M | H | L | H | M | H ||

6. Make sure that when mapping out channels to the sequence, you choose a worst-case slot to start the mapping. For example, when estimating WCL for a high-priority channel, do not start the mapping in the last high-priority slot in a seven-slot sequence, as that is a best case for a high-priority channel since another high-priority time slot is next.
7. Instead of always using the longest thread in the function as the worst-case thread, evaluate the threads in the function that will be used in the system and use the appropriate worst-case threads. For example, in the preceding example of first-pass analysis, the PWM was shown to be able to achieve a high time and low time of 2475 ns under worst-case conditions. This was derived using the longest PWM thread of 24 CPU clocks. This longest thread is actually thread 2, the thread that is entered after the pin has just gone high. Thread 3, the thread that is entered after the pin has just gone low, requires only 2 CPU clocks. Therefore, in the first-pass example, the high time was correctly derived, but the low time is actually shorter than was estimated.

37.16.5.5.2 Second-pass analysis example

This example requires three 50% PWM waveforms: one 5 kHz (200 ms/period) and two 50 kHz (20 ms/period), each running DC motors. (Remember that the PWM function requests service from the eTPU after each high time and after each low time, so the eTPU must handle a request every 100 ms for the 5 kHz PWM and every 10 ms for the 50 MHz PWM.)

Note

This example uses square waves for simplicity. Notice that to use a PWM waveform in the typical way, in which the pulse is modulated, the pulse must not be modulated in a way that violates the worst-case latency requirements.

This example also uses one DIO channel monitoring a signal level every millisecond and one PPWA channel in mode 0 monitoring the speed of the 5-kHz DC motor. The PPWA must measure periods of 5 kHz (200 ms/period).

The CPU is interrupted by the channel running the PPWA function after measuring 200 periods (every 40 ms). The interrupt service routine performs an averaging of the period accumulation and checks it against a known parameter. The interrupt service time is so short and infrequent that it is a tiny fraction of total system time. The interrupt service routine contains no polling of the SDM. Therefore a realistic RCR = 0%.

37.16.5.5.2.1 First-Try System Configuration

Try a system configuration that seems likely to work. If it does not, change priority levels or channel numbers.

The 5 kHz and 50 kHz PWMs are the most time-critical functions. Those are assigned high priority. PPWA is assigned middle priority. The DIO is low performance and is assigned low priority. Refer to the following table.

Table 37-426. First-Try System Configuration

Channel	Priority	Function ^{1, 2}
0	High	PWM at 50 kHz (needs a 4-μs WCL)
1	High	PWM at 50 kHz (needs a 4-μs WCL)
2	High	PWM at 5 kHz (needs a 40-μs WCL)
8	Middle	PPWA at 5 kHz (needs a 80-μs WCL)
15	Low	DIO as input at rate of 1 ms

1. 0% RAM collision rate

2. CPU clock rate = 40 MHz, or 60 ns per clock period

With this system configuration, worst-case service time for each active channel is determined as follows:

1. Longest thread of PWM is 24 CPU clocks with four RAM accesses.

$$24 + ((4 \text{ RAM accesses} + 1) * 0 * 2 \text{ CPU clock waits}) = 24 \text{ CPU clocks}$$

Channels 0-2 worst-case service time = 24 CPU clocks.

2. Longest thread of PPWA in mode 0 is 44 CPU clocks with nine RAM accesses.

$$44 + ((9 \text{ RAM accesses} + 1) * 0 * 2 \text{ CPU clock waits}) = 44 \text{ CPU clocks}$$

Channel 8 worst-case service time = 44 CPU clocks.

3. Longest thread of DIO is ten CPU clocks with four RAM accesses.

$$10 + ((4 \text{ RAM accesses} + 1) * 0 * 2 \text{ CPU clock waits}) = 10 \text{ CPU clocks}$$

Channel 15 worst-case service time = 10 CPU clocks.

To find the WCL for channel 0, assume channel 0 has just finished service.

Map the channels in the H-M-H-L-H-M-H sequence. See the following figure.

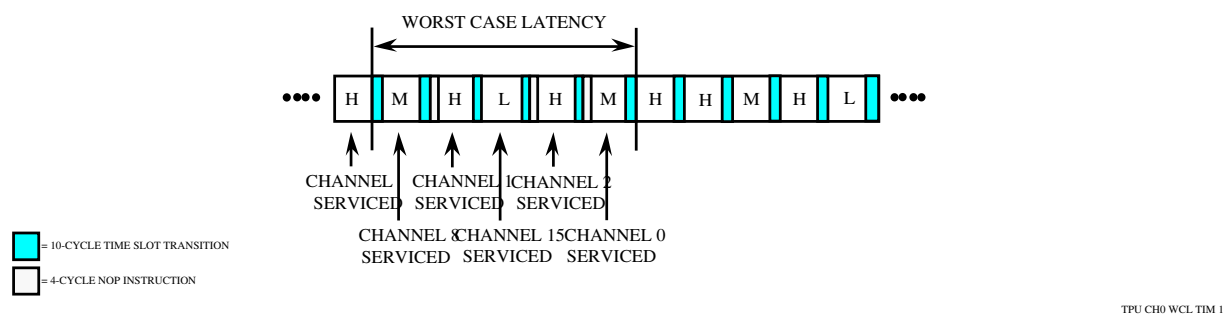


Figure 37-343. Worst-Case Latency for Channel 0 (First Try)

Conclusion: with this system configuration, worst-case latencies for channels 0 and 1 are too high (WCL for channel 1 is the same as WCL for channel 0). Try a different system configuration.

37.16.5.5.2.2 Second-Try System Configuration

The second-try system configuration is shown in the following table.

Table 37-427. Second-Try System Configuration

Channel	Priority	Function ^{1, 2}
0	High	PWM at 50 kHz (needs a 4-μs WCL)
1	High	PWM at 50 kHz (needs a 4-μs WCL)
2	Middle	PWM at 5 kHz (needs a 40-μs WCL)
8	Middle	PPWA at 5 kHz (needs a 80-μs WCL)
15	Low	DIO as input at rate of 1 ms

- 1. 0% RAM collision rate
- 2. CPU clock rate = 40 MHz, or 60 ns per clock period

To find the WCL for channel 0, assume channel 0 has just finished service. Map the channels in the H-M-H-L-H-M-H sequence. See the following figure.

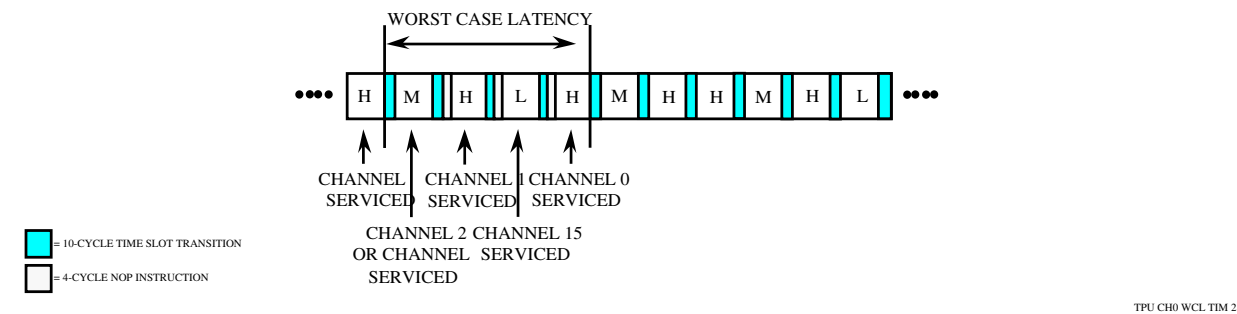


Figure 37-344. Worst-Case Latency for Channel 0 (Second Try)

Conclusion: with this system configuration, the WCL of both channel 0 and channel 1 is 3.85 ms, which is within the limit of 4 ms needed for a 50-kHz PWM.

Next, find the WCL for channel 2. Assume channel 2 has just finished service. Map the channels in the H-M-H-L-H-M-H sequence. See the following figure.

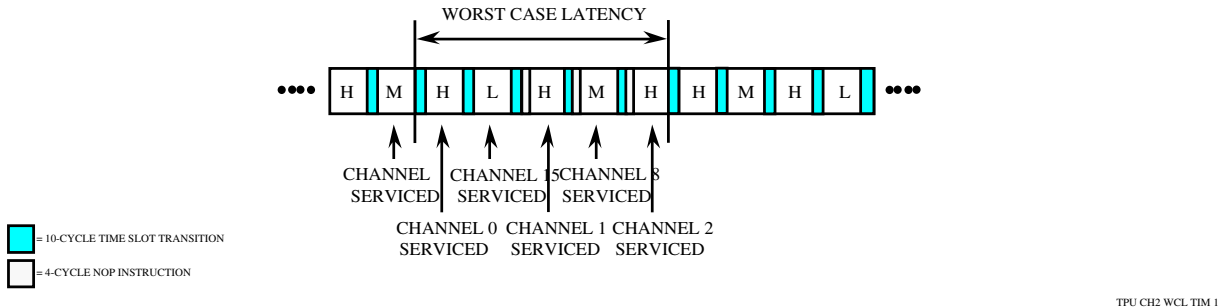


Figure 37-345. Worst-Case Latency for Channel 2

Conclusion: with this system configuration, the WCL for channels 2 and 8 is 4.7 ms, which is within the 40 and 80 ms WCL requirements.

Notice that channels 2 and 8 are well within their WCL requirements. The system could be reconfigured as shown in the following table to give channels 0 and 1 a larger margin while still keeping channels 2, 8 and 15 within their WCL requirements.

Table 37-428. Second-Try System with Channel 0 and 1 Reconfigured

Channel	Priority	Function ^{1, 2}
0	High	PWM at 50 kHz (needs a 10- μ s WCL)
1	High	PWM at 50 kHz (needs a 10- μ s WCL)
2	Middle	PWM at 5 kHz (needs a 40- μ s WCL)
8	Low	PPWA at 5 kHz (needs a 80- μ s WCL)
15	Low	DIO as input at rate of 1 ms

1. 0% RAM collision rate

2. CPU clock rate = 40 MHz, or 60 ns per clock period

37.16.6 Endianness

The following table shows the parameter addresses for big and little endian machines.

Table 37-429. Parameter Addresses and Endianness

Parameter	byte address offset (n = word address offset)	
	big endian	little endian
32-bit	4*n	
24-bit	4*n + 1	4*n
32-bit parameter's most significant byte	4*n	4*n + 3
24-bit parameter's most significant byte	4*n + 1	4*n + 2
least significant byte	4*n + 3	4*n

37.16.7 Memory Error Service Procedures

Follows separate procedures for correctable and non-correctable error interrupt service routines. It is assumed that the non-correctable error ISR has higher priority, and cannot be preempted by the correctable error ISR. For memory error support details, see [Memory error support](#).

Note

The memory error support hardware does not (and is not intended to) guarantee that no error occurred will go unreported. However, combined with the procedures below, the information loss is minimized to statistical irrelevance on most scenarios.

37.16.7.1 Non-correctable error service procedure

The basic procedure below can be followed in a Global Exception or a bus error machine check service routine. It handles the service for both SCM and SDM errors, as they are symmetrical at this level. However, a real implementation is likely to split the procedure in two, one for each memory.

1. Confirm/identify the error source, checking assertion of the [eTPU Module Configuration Register \(eTPU_SCR_MCR\)](#) fields SCMERR and SDMERR.
2. Save the state(s) of [eTPU Module Configuration Register \(eTPU_SCR_MCR\)](#) fields CRDIS/DRDIS.
3. Set the [eTPU Module Configuration Register \(eTPU_SCR_MCR\)](#) CRDIS/DRDIS=1 to disable further updates to the respective report registers, guaranteeing coherency.

4. Check whether the report registers were updated by a correctable or non-correctable error; to do this, check the [eTPU Code Error Report Syndrome Register \(eTPU_MESR_CERSR\)](#) syndrome (for SCM) and/or the [eTPU Data Error Report Syndrome Register \(eTPU_MESR_DERSR\)](#) syndromes (for SDM, only those with the respective BE flags asserted). If it is a correctable error report, skip the steps 5 and 6 for the respective memory.
5. Identify the error source (host, microengine or MISC/CDC) by checking the field ERR_ACC in the register(s) [eTPU Code Error Report Address Register \(eTPU_MESR_CERAR\)](#) and [eTPU Data Error Report Address Register \(eTPU_MESR_DERAR\)](#); take or schedule the appropriate application-dependent recover actions.
6. Update error statistics as needed; it may include recording the access source, address, data and syndrome from:
 - [eTPU Code Error Report Address Register \(eTPU_MESR_CERAR\)](#)
 - [eTPU Data Error Report Address Register \(eTPU_MESR_DERAR\)](#)
 - [eTPU Code Error Report Data Register \(eTPU_MESR_CERDR\)](#)
 - [eTPU Data Error Report Data Register \(eTPU_MESR_DERDR\)](#)
 - [eTPU Code Error Report Syndrome Register \(eTPU_MESR_CERSR\)](#)
 - [eTPU Data Error Report Syndrome Register \(eTPU_MESR_DERSR\)](#)
7. Save the status of non-correctable error overflows flagged in the [eTPU Memory Error Status Register \(eTPU_MESR_MESR\)](#) fields CNCOVR/DNCOVR.
8. Set the [eTPU Memory Error Control Register \(eTPU_MESR_MECR\)](#) fields CRDIS/DRDIS to the state(s) saved in the step 2. That assures the access coherency of a preempted Correctable Error Service Procedure (which may have set CRDIS/DRDIS before this procedure) is not disturbed.
9. Clear the [eTPU Memory Error Status Register \(eTPU_MESR_MESR\)](#) fields CNCOVR/DNCOVR and CNCERR/DNCERR at once, in a single write to eTPU_MESR_MESR.
10. Use the overrun status saved on step 7 to update the error statistics, as needed.
11. Check the other Global Exception flags in [eTPU Module Configuration Register \(eTPU_SCR_MCR\)](#); take or schedule the appropriate (application dependent) recover actions.
12. Clear the Global Exception, programming [GEC](#) =1.

37.16.7.2 Correctable error service procedure

The basic procedure below can be followed in a Memory Error Interrupt service routine (see [Interrupts and data transfer requests](#)). It handles the service for both SCM and SDM errors, as they are symmetrical at this level. However, a real implementation is likely to split the procedure in two, one for each memory.

1. Confirm/identify the error source, checking assertion of the [eTPU Memory Error Status Register \(eTPU_MESR_MESR\)](#) fields CCERR/DCERR.
2. Program the [eTPU Memory Error Control Register \(eTPU_MESR_MECR\)](#) fields CRDIS=1 and/or DRDIS=1, to disable further updates to the respective report registers, guaranteeing coherency.
3. Check whether the report registers were updated by a correctable or non-correctable error; to do this, check the [eTPU Code Error Report Syndrome Register \(eTPU_MESR_CERSR\)](#) syndrome (for SCM) and/or the [eTPU Data Error Report Syndrome Register \(eTPU_MESR_DERSR\)](#) syndromes (for SDM, only those with the respective BE flags asserted). If it is a non-correctable error report, skip the steps 4 and 5 for the respective memory.
4. Identify the error source (host, microengine or MISC/CDC) by checking the field ERR_ACC in [eTPU Code Error Report Address Register \(eTPU_MESR_CERAR\)](#) and [eTPU Data Error Report Address Register \(eTPU_MESR_DERAR\)](#); take or schedule the appropriate application-dependent recover actions, which may include soft-error fixing (see [Error fixing](#)).
5. Update error statistics as needed; it may include recording the access source, address, data and syndrome from:
 - [eTPU Code Error Report Address Register \(eTPU_MESR_CERAR\)](#)
 - [eTPU Data Error Report Address Register \(eTPU_MESR_DERAR\)](#)
 - [eTPU Code Error Report Data Register \(eTPU_MESR_CERDR\)](#)
 - [eTPU Data Error Report Data Register \(eTPU_MESR_DERDR\)](#)
 - [eTPU Code Error Report Syndrome Register \(eTPU_MESR_CERSR\)](#)
 - [eTPU Data Error Report Syndrome Register \(eTPU_MESR_DERSR\)](#)
6. Save the status of correctable error overflows flagged in the [eTPU Memory Error Status Register \(eTPU_MESR_MESR\)](#) fields CCOVR/DCOVR.
7. Program the [eTPU Memory Error Control Register \(eTPU_MESR_MECR\)](#) fields CRDIS/DRDIS=0.
8. Clear the [eTPU Memory Error Status Register \(eTPU_MESR_MESR\)](#) fields CCOVR/DCOVR and CCERR/DCERR at once, in a single write to eTPU_MESR_MESR.

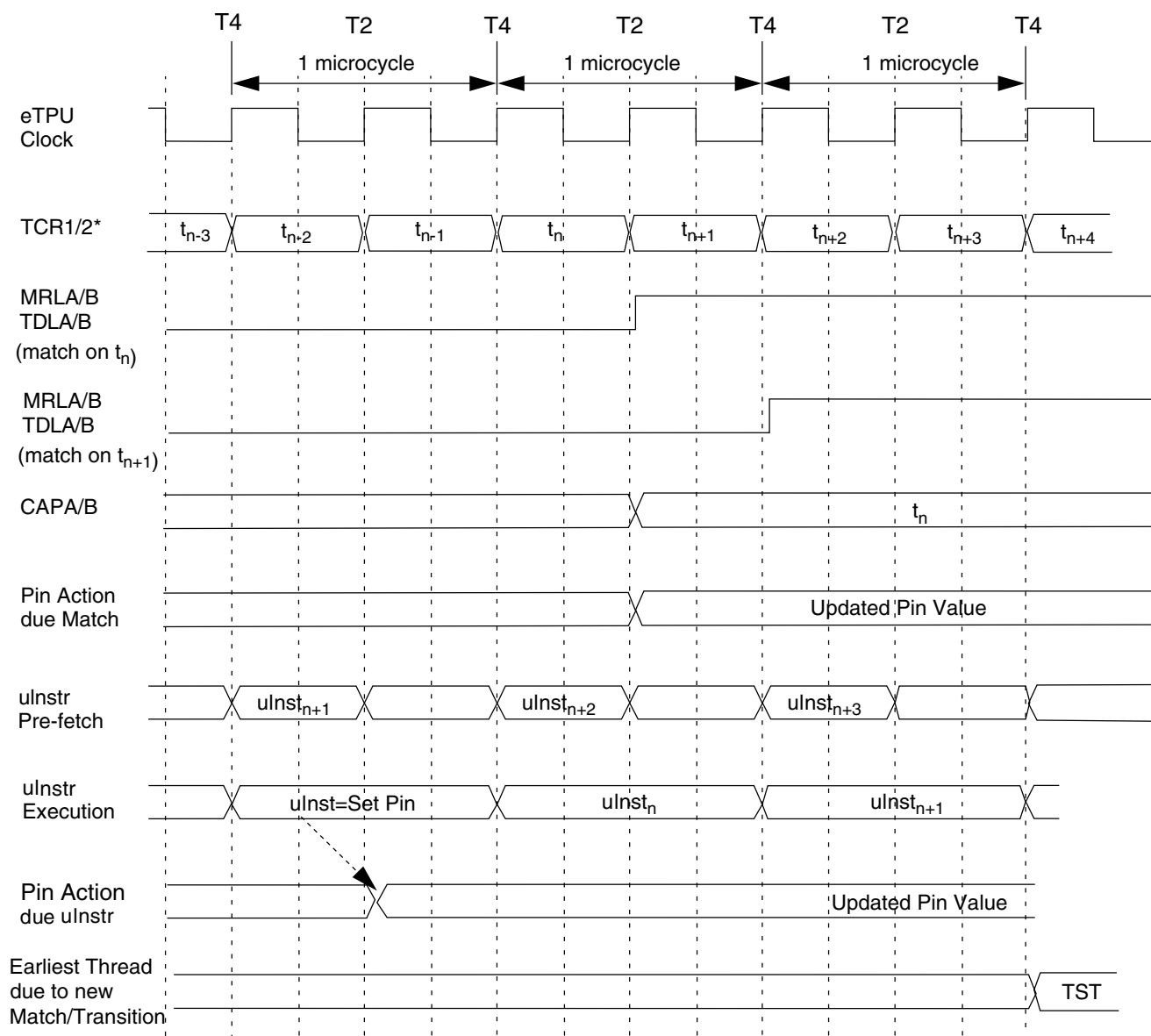
9. Use the saved overrun status to update the error statistics, as needed.

37.17 Appendices

37.17.1 Microcycle and I/O timing

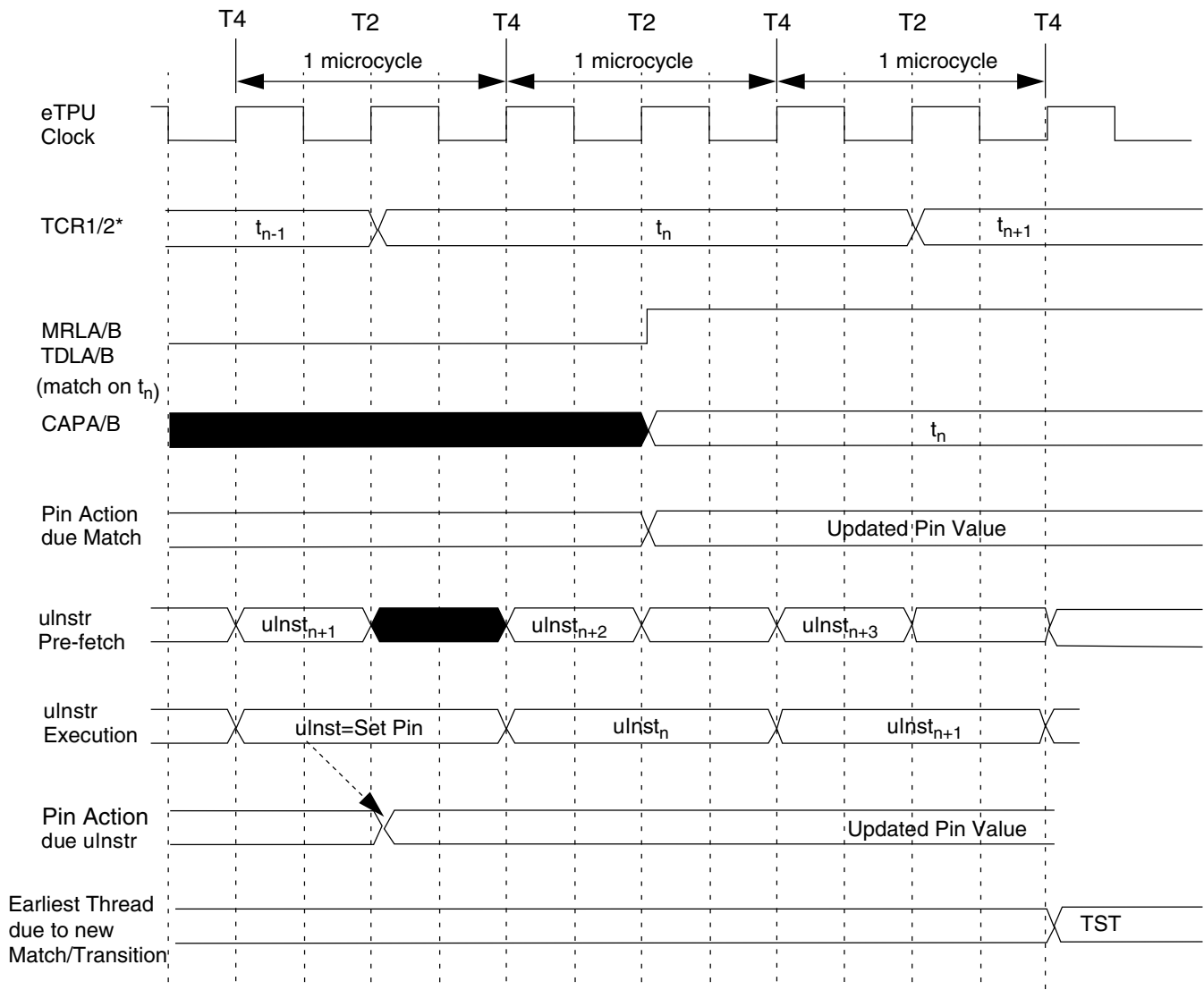
37.17.1.1 Execution and channel timing

The following figure shows the main timings related to microinstruction execution when channels and timebases run on T2 timing.



Note: *TCR clock/prescaler selection = 1 x eTPU clock

Figure 37-346. Execution, timebase and channel T2 timing



Note: *TCR clock/prescaler selection = 4x eTPU clock

Figure 37-347. Execution, timebase and channel T2/T4 timing

The sequential occurrence of the four T states (T1-T4) constitutes a microcycle. Only T2 and T4 are taken as reference for timings, either internal or external. T2 and T4 have the timing of the positive eTPU clock pulses, and are used in most of the eTPU logic in an edge-triggered design style.

Two additional T states are derived from the eTPU clocks: $\overline{T2}$ and $\overline{T4}$. $\overline{T2}$ occurs when the eTPU loses SDM arbitration to a bus master. $\overline{T4}$ occurs in halt state (due to a breakpoint or device debug request assertion, for instance), or in stall state (due to an NDEDI queue full); see [Development support features](#) for more details.

$\overline{T2}$ and $\overline{T4}$ states are defined as microcycle timing states (not to be confounded with logic states) of one eTPU clock in which the T clocks continue to run, but the control signals associated with the clocks are unaffected. That is, no operation occurs during these states.

Both $\overline{T2}$ and $\overline{T4}$ states occur in multiples of two eTPU clocks to keep the microengine synchronized with the free running channels, which are unaffected and keep on working as in T2 and T4.

Thus, the eTPU has two types of timing states:

- $\overline{T2}$: hold execution for SDM access, from clock pulse T2 until one of the next T2 clock pulses of another microcycle.
- $\overline{T4}$: hold execution in debug mode or stall, from clock pulse T4 until one of the next T4 clock pulses of another microcycle.

The two following figures show the timing of $\overline{T2}$ and $\overline{T4}$ timing states, respectively.

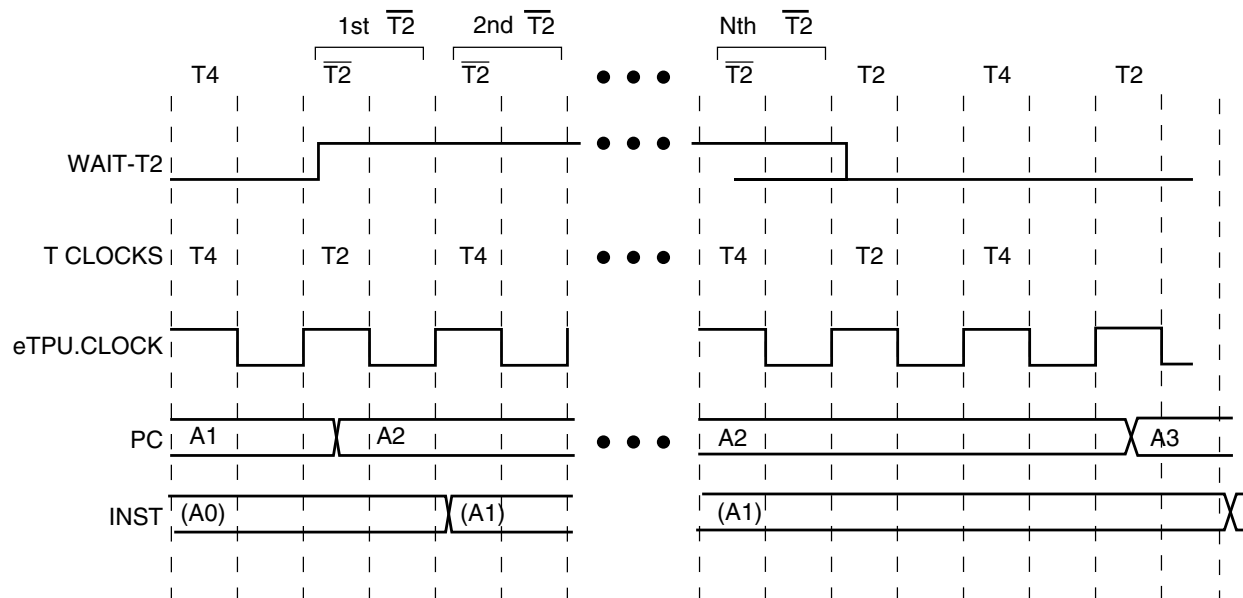
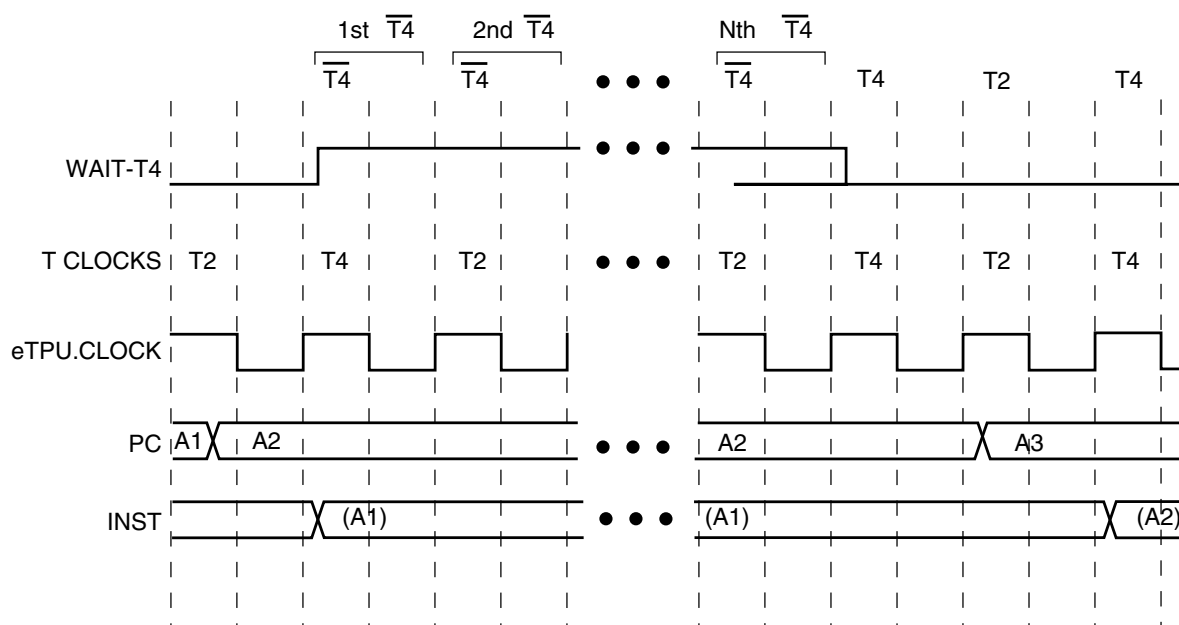


Figure 37-348. $\overline{T2}$ timing

Figure 37-349. $\overline{T4}$ timing

37.17.1.2 Input/Output Signal Delays

The synchronizer, filter and edge detection logic delay the input signal transitions.

The Filter Delay varies with the filter clock ([eTPU Engine 1 Configuration Register \(eTPU_SCR_ECR_ENG1\)](#) and [eTPU Engine 2 Configuration Register \(eTPU_SCR_ECR_ENG2\)](#) field FPSCK) and the filter mode used, as shown in the [Table 37-355](#). For any given transition, it depends on the phase of the filter clock when the input transition happens. In integration mode (TCRCLK filtering only), it also depends on the state of the integrator counter. The Total Delay is defined as the number of eTPU clock rising edges between the input transition and the setting of TDLA/B, TCR1/2 incrementing, or EAC tooth sensing (TCRCLK) in angle mode. The synchronizer delay is 2 or 3 eTPU clocks, depending on the phase of the synchronizer when the input transition happens. The edge detection takes 1 more eTPU clock. The total delays are, thus:

```

Min. Total Delay = Min. Synchronizer Delay + Min. Filter Delay + Edge Detection Delay
Min. Total Delay = 3 + Min. Filter Delay
Max. Total Delay = Max. Synchronizer Delay + Max. Filter Delay + Edge Detection Delay
Max. Total Delay = 4 + Max. Filter Delay

```

The channel filters can be bypassed, so nullifying the filter delays in the equations above.

See the chip-specific eTPU information for connections between eTPU and input glitch filters.

The channel output flip-flops drive the eTPU output signals directly, without any synchronous delays.

37.17.2 Channel Mode Summary

The initial state considered for all submodes is channel flags MRLEA and MRLEB asserted, MRLA, MRLB, TDLA and TDLB negated. From initial state one can follow the table below and verify how each submode behaves in a determined sequence of events. Note that the actions performed by an event type depend on all previous events following the initial state, for a given channel submode.

There are three columns for each event: one for event type, one for enable/disable actions and one for capture. **Event type** column can be matchA, matchB, transA and transB (for double transition modes). **Enable/disable actions** column (identified as "[blocks] (enables)" in column head) specifies which other events are enabled or disabled. Initially disabled events (specified in "**initially blocked**" column) are usually enabled by other events.

In double transition submodes, the first transition detected is always considered transA and the second is considered transB. This means that transA event actually enables the detection of transB event. This is not explicit in the table, since it is a general behavior for all double transition submodes.

A sequence of four events (two matches and two transitions) are necessary to describe the behavior of some channel submodes. When a determined sequence of events has less than four events, the other event columns are left blank.

Cells in an "**event type**" column that have light-grayed background indicate that a service request is generated. More than one event in the same event sequence can issue service request.

Note

The two tables below do not exhaust all possibilities of channel logic event sequences, because it does not account for possible microcode interventions. For instance, if matches are blocked

by first transition and microcode resets TDLA, the matches become enabled again, and from this point on the channel behaves as if the first transition had never occurred.

Table 37-430. Channel Mode Summary 1 of 2

Mode	Initially Blocked	1st Event			2nd Event		
		Event Type	[Blocks] (Enables) ¹	Capture	Event Type	[Blocks] (Enables)	Capture
em_nb_st	none	match 1/2 ²	none	1/2	match 2/1	none	2/1
					trans 1	[matches]	both
		trans 1	[matches]	both	trans 2	none	2
em_nb_dt	none	match 1/2	none	1/2	match 2/1	none	2/1
		match 1	none	1	trans 1	none	1
		match 2	none	2	trans 1	[match 1]	1
		trans 1	[match 1]	1	match 2	none	2
					trans 2	[match 2]	2
em_b_st	none	match 1	[match 2]	both	trans 1	[matches]	both
		match 2	[match 1]	both			
		trans 1	[matches]	both	trans 2	none	2
em_b_dt	none	match 1	[match 2]	both	trans 1	none	1
		match 2	[match 1]	both			
		trans 1	[match 1]	1	match 2	none	2
					trans 2	[match 2]	2
bm_st	none	match 1/2	none	1/2	match 2/1	none	2/1
					trans 1	[matches]	both
		trans 1	[matches]	both	trans 2	none	2
bm_dt	none	match 1/2	none	1/2	match 2/1	none	2/1
					trans 1	none	1
		trans 1	none	1	match 1/2	none	none/2
					trans 2	[matches]	2
m2_st	trans 1	match 1	(trans 1)	1	match 2	none	2
					trans 1	[matches]	both
		match 1 and match 2	(trans 1)	both	trans 1	[matches]	both
		match 2	[match 1]	2	n/a	n/a	n/a
m2_dt	trans 1	match 1	(trans 1)	1	trans 1	none	1
					match 2	none	2
		match 1 and match2	(trans 1)	both	trans 1	none	1
		match 2	[match 1]	2	n/a	n/a	n/a
m2_o_st	trans 1 match 2	match 1	(match 2) (trans 1)	1	trans 1	[matches]	both
					match 2	[trans 1]	2

Table continues on the next page...

Table 37-430. Channel Mode Summary 1 of 2 (continued)

Mode	Initially Blocked	1st Event			2nd Event		
		Event Type	[Blocks] (Enables) ¹	Capture	Event Type	[Blocks] (Enables)	Capture
m2_o_dt	trans 1 match 2	match 1	(match 2) (trans 1)	1	trans 1	none	1
		match 2			match 2	[trans 1]	2
sm_st ³	match 2	match 1	none	both	trans 1	none	both
		trans 1	[match 1]	both	trans 2	none	2
sm_dt	match 2	match 1	none	both	trans 1	none	1
		trans 1	none	1	match 1	none	2
		trans 1	none	1	trans 2	[match1]	2
sm_st_e ⁴	match 2 trans 2	match 1	none	1	trans 1	none	1
		trans 1	[match 1]	1	n/a	n/a	n/a

1. Transition 1 always enables Transition 2.
2. Shading indicates that a service request is generated.
3. sm_st is compatible with TPU3 channel logic.
4. It is not possible to include all functionality of this submode in table.

Table 37-431. Channel Mode Summary 2 of 2

Mode	Initially Blocked	3rd Event			4th Event		
		Event Type	[Blocks] (Enables) ¹	Capture	Event Type	[Blocks] (Enables)	Capture
em_nb_st	none	trans 1 ²	[matches]	both	trans 2	none	2
					n/a	n/a	n/a
		trans 2	none	2	trans 2	none	2
em_nb_dt	none	trans 1	none	1	trans 2	none	2
		match 2	none	2	trans 2	none	2
		trans 2	[match 2]	2	n/a	n/a	n/a
		trans 2	[match 2]	2	n/a	n/a	n/a
		trans 2	none	2	n/a	n/a	n/a
em_b_st	none	trans 2	none	2	n/a	n/a	n/a
em_b_dt	none	trans 2	none	2	n/a	n/a	n/a
		trans 2	none	2	n/a	n/a	n/a
bm_st	none	trans 1	[matches]	both	trans 2	none	2
		trans 2	none	2	n/a	n/a	n/a
bm_dt	none	trans 1	none	1	trans 2	[matches]	2
		match 2/1	none	2 / none	trans 2	[matches]	2
		trans 2	[matches]	2	n/a	n/a	n/a
		match 2/1	none	2 / none	trans 2	[matches]	2
		trans 2	[matches]	2	n/a	n/a	n/a

Table continues on the next page...

Table 37-431. Channel Mode Summary 2 of 2 (continued)

Mode	Initially Blocked	3rd Event			4th Event		
		Event Type	[Blocks] (Enables) ¹	Capture	Event Type	[Blocks] (Enables)	Capture
m2_st	trans 1	trans 1	[matches]	both	trans 2	none	2
		trans 2	none	2	n/a	n/a	n/a
		trans 2	none	2	n/a	n/a	n/a
m2_dt	trans 1	trans 2	[match 2]	2	n/a	n/a	n/a
		match 2	none	2	trans 2	none	2
		trans 1	none	1	trans 2	none	2
		trans 2	none	2	n/a	n/a	n/a
m2_o_st	trans 1 match 2	trans 2	none	2	n/a	n/a	n/a
m2_o_dt	trans 1 match 2	trans 2	[match 2]	2	n/a	n/a	n/a
		match 2	[trans 2]	2	n/a	n/a	n/a
sm_st ³	match 2	trans 2	none	2	n/a	n/a	n/a
sm_dt	match 2	trans 2	none	2	n/a	n/a	n/a
		trans 2	none	2	n/a	n/a	n/a
sm_st_e ⁴	match 2 trans 2	n/a	n/a	n/a	n/a	n/a	n/a

1. Transition 1 always enables Transition 2.
2. Shading indicates that a service request is generated.
3. sm_st is compatible with TPU3 channel logic.
4. It is not possible to include all functionality of this submode in table.

37.17.3 MISC Algorithm

The MISC generator is based on the following polynomial:

$$G(x) = 1 + x^1 + x^2 + x^{22} + x^{31} \text{ (equivalent to feedback mask = 0x80400007)}$$

The MISC signature generation starts by clearing the MISC Accumulator value to 0 and preloading the MISC Counter with the highest SCM address. It then steps through each address decrementing the counter, reading 32 bit values and following the algorithm below:

```

If the least significant bit in MISC is 1 then
    MISC = MISC right shifted by 1 bit
    MISC = MISC XOR 0x80400007
else
    MISC = MISC right shifted by 1 bit
end if
MISC = MISC XOR RAM data

```

The code example below shows an excerpt of C code that calculates the MISC signature for a given array of data, based on the previous algorithm:

Appendices

```
#define SCM_size    (MAX_SCM_ADDRESS / 4)    /* last byte address - converted to 32-bit word */
#define POLY        0x80400007                /* G(x) = 1 + x1 + x2 + x22 + x31 */
/*****
FUNCTION : void calc_misc()
PURPOSE : This function calculates the MISC value.
INPUTS NOTES : none
RETURNS NOTES : MISC value
GENERAL NOTES : the array 'unsigned int data[]' represents the actual memory
                  array, organized in 32-bit words.
*****/
unsigned int calc_misc (void)
{
    int j; /* loop counter */
    unsigned int misc = 0;
    for (j = (SCM_size-1); j >= 0 ; j--) {    /* SCM_size has the number of 32-bit words in SCM
*/
        if (misc & 0x1) {
            misc >>= 1;
            misc ^= POLY;
        }
        else {
            misc >>= 1;
        }
        misc ^= data[j];                    /* data[j] is the actual 32-bit word taken from the SCM
array
*/
    }
    return (misc);                        /* final signature calculated */
};
```

The value calculated by this algorithm must be loaded into [eTPU MISC Compare Register \(eTPU_SCR_MISCCMPR\)](#) prior to activating the SCM MISC calculator in eTPU. Once the MISC calculator is activated (bit SCMMISEN in register [eTPU Module Configuration Register \(eTPU_SCR_MCR\)](#) is written to 1) eTPU itself will start this procedure⁴² reading the SCM whenever allowed by microengine. At the end of the cycle, when all the array has been read and the SCM signature is calculated, the Host CPU can be notified via Global Exception if the MISC Accumulator does not match the value in eTPU_SCR_MISCCMPR.

The average time taken by MISC to complete the signature of the whole SCM can be given by the formula:

$$\text{Average MISC period} = S / (4 * F * (1 - L))$$

where f is clock frequency, S is SCM size in bytes and L is eTPU load (as a percentage of execution clocks over a period of time, including TST clocks).

Further detail on MISC calculation can be found on [SCM Test - Multiple Input Signature Calculator](#). The application note **AN2192 - Detecting Errors in the Dual Port RAM (DPTRAM) Module** is also a good source of information (although it refers to TPU) on MISC signature.

42. eTPU MISC hardware is optimized to read 32-bit words from memory and to calculate this CRC in parallel, rather than shifting one bit at a time. The actual implementation inside eTPU, although bringing to the same results, does not match exactly the algorithm shown here.

Chapter 38

Reaction Module 2 (REACM2)

38.1 Chip-specific REACM2 information

This device contains one REACM2 module with 10 Reaction Channels. It is on the eTPU clock domain.

38.1.1 REACM2 interaction with other modules

The following figure shows the interaction of the Reaction Module 2 with other modules on the chip at a high level.

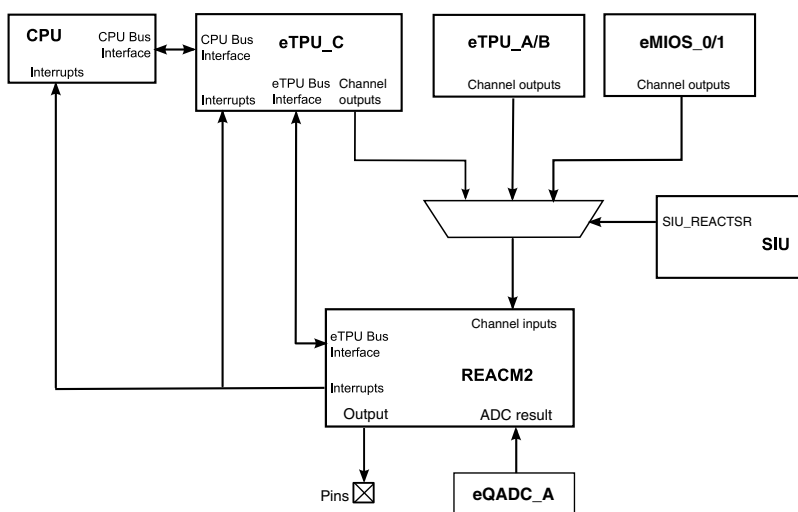


Figure 38-1. REACM2 interaction with other modules

- eTPU_C, a co-processor designed for timing control, has an interface with REACM2 that allows eTPU_C to read and write data to/from REACM2 and to read and write REACM2 registers.
- All accesses to REACM2 registers, including by the CPU, occur through eTPU_C.
- The REACM2 sends interrupt signals to both eTPU_C and INTC.
- Timer trigger inputs are multiplexed from eTPU_C, eTPU_A/B, and eMIOS_0/1.

- From its two internal ADCs (ADC0/1), eQADC_A sends ADC conversion results to the REACM2. See [Routing conversion results to REACM2 and DECFILTER modules](#) for details.
- The reaction channels' output is sent to output pins.

38.1.2 Timer inputs

The REACM2 timer input channels receive trigger input signals from eTPU or eMIOS. The timer input multiplexer is controlled by the SIU Reaction Module Trigger Selection Register (SIU_REACTSR).

Table 38-1. REACM2 timer inputs

REACM2 timer input channel	eTPU_A	eTPU_B	eTPU_C / eMIOS_0	eMIOS_0 / eMIOS_1
0	eTPU_A ch.10	eTPU_B ch.22	eTPU_C ch.13	eMIOS_0 ch.16
1	eTPU_A ch.11	eTPU_B ch.23	eTPU_C ch.14	eMIOS_0 ch.17
2	eTPU_A ch.12	eTPU_B ch.24	eTPU_C ch.15	eMIOS_0 ch.18
3	eTPU_A ch.13	eTPU_B ch.25	eTPU_C ch.16	eMIOS_0 ch.19
4	eTPU_A ch.14	eTPU_B ch.10	—	eMIOS_0 ch.0
5	eTPU_A ch.15	eTPU_B ch.11	—	eMIOS_0 ch.1
6	eTPU_A ch.16	eTPU_B ch.12	—	eMIOS_0 ch.2
7	eTPU_A ch.17	eTPU_B ch.13	—	eMIOS_0 ch.3
8	eTPU_A ch.18	eTPU_B ch.14	eTPU_C ch.28	eMIOS_1 ch.4
9	eTPU_A ch.19	eTPU_B ch.15	eTPU_C ch.29	eMIOS_1 ch.5
10	eTPU_A ch.20	eTPU_B ch.16	eTPU_C ch.30	eMIOS_1 ch.6
11	eTPU_A ch.21	eTPU_B ch.17	eTPU_C ch.31	eMIOS_1 ch.7
12	eTPU_A ch.22	eTPU_B ch.18	eMIOS_0 ch.4	eMIOS_1 ch.0
13	eTPU_A ch.23	eTPU_B ch.19	eMIOS_0 ch.5	eMIOS_1 ch.1
14	eTPU_A ch.24	eTPU_B ch.20	eMIOS_0 ch.6	eMIOS_1 ch.2
15	eTPU_A ch.25	eTPU_B ch.21	eMIOS_0 ch.7	eMIOS_1 ch.3

The timer input channel used by a Reaction Channel is determined by REACM2_CHRRn[CHIR].

38.1.3 REACM2 buffer overflow prevention

When two samples are feeding REACM2, the REACM2 buffer can overflow. The overflow can occur if two parallel conversions (ADC0/1 in EQADC_A) routed to REACM2 finish at approximately the same time.

To prevent the overflow, program the timing of the triggers (samples feeding REACM2) so they occur at least 6 REACM2 clock periods apart.

38.2 Introduction

The REACM2 is composed of 10 reaction channels. Each channel contains three outputs. Its main application is in the area of solenoid control for direct injection systems, valve control in automatic transmissions, and other current control applications. It is connected to the on-chip ADC which monitors the current on the solenoid or valve. Based on that the reaction channel generates an output signal that modulates the current circulating in the solenoid or valve. It is a cost-effective solution due to extensive sharing of several resources among channels and parameterized register banks for adequate dimensioning of resources and functionality.

38.2.1 Features

The REACM2 features include:

- Per channel architecture for independent output control
- Interface with on-chip ADC for fast response times
- Interfaced to timer modules
- Hardware connection with on-chip timer channels with channel routing capability
- Shared Modulation Control bank
- Dynamic timer allocation
- Three outputs per channel to support different driver architectures
- Flexibility to operate based on timing and threshold
- On-the-fly capture of ADC result reference for fast calibration
- Open and short circuit monitoring capability

38.2.2 Modes of Operation

Operation modes of this module are described below.

38.2.2.1 Programming Mode

After reset, the Reaction module is in the Programming mode. In this mode all channels are disabled and outputs are at logic zero. Note that this state does not necessarily mean that zero is the neutral state for the channel load, thus care must be taken in order to disconnect the channel from the load in this case.

In the programming mode the host CPU writes all module parameters including:

1. Modulation word data
2. Channel control data
3. Threshold data
4. Timer Bank data
5. Hold (on/off) Timer bank data
6. Timer router data
7. ADC router data
8. Period Shift Delay time
9. Period Generator timers

The last action to perform is to *enable* the channel. After that the channel is able to respond to timer signals and ADC data, thus able to perform modulation on the output pins. It is recommended to keep the timer signals inactive until all data to all reaction channels are programmed and all channels have been put in the *enabled* mode.

38.2.2.2 Low Power Mode

Coming out from reset all channels are in the disabled state. The channel may also be in low power mode depending on an instantiation parameter that configures the initial state of the MDIS in the Module Configuration Register. If MDIS=1 the module clock may be disabled, allowing for a low power state. The low power mode is controlled either by the MDIS bit or by a global stop signal. There is no explicit clock gating implemented in hardware within the Reaction Module. Reaction module clock gating is implemented in the SoC. Low power mode must be entered only when all channels are disabled by CHEN=00.

38.2.2.3 Channel Modes

After a channel is put in *enabled* mode that channel is also said to be in the *normal* mode of operation, which means it responds to timer signals from the timer inputs connected to the Reaction Module and also to ADC results received from the on-chip ADC module. Channel outputs are controlled in accordance with those inputs in order to perform an output modulation process. When performing a modulation the Reaction Channel is said to be in the *active state*. The Reaction Channel modes are described below in relation to the ability to execute a modulation:

- Disabled: the channel cannot execute modulation.
- Enabled: the channel is able to execute a modulation. It may be in the *Active* or *Inactive* states.
 - Inactive state: the channel is not executing a modulation.
 - Active state: the channel is executing a modulation.

38.2.2.4 Debug Mode

The Reaction Module Debug operation is defined by bits FRZ and FREN in the Module Configuration Register. In debug mode all Timers are freezed (halted), including the timers in the Shared Time Bank and Hold (on/off) timers. The module can enter debug mode in two different ways, either by software control or by the hardware debug input signal controlled by the chip logic. In both cases the Reaction Module enters debug mode only if enabled by the FREN bit:

- If both FREN and FRZ bits are set, the module enters debug mode.
- If FREN bit is set and a global debug signal is asserted, the module enters debug mode.

In debug mode, the channel outputs are held at HOD, LOD, or DOFF state, as determined by the channel current state. When resuming normal operation after exiting debug mode, the channel performs the read of the first Mod Word and the channel output is set to DOFF until the next timer control rising edge occurs.

MAXL flag is the only flag that operates in debug mode. All other flags retain the present state while the module is in debug mode. Note that the corresponding error flag in the Global Error Flag register is also set.

The ADC Input Sensor Input Register allows direct access for write to the TAG and ADC result values input to the Reaction Module. This software control may be used for module debug purposes.

38.2.3 Block Diagram

Figure 38-2 shows the on-chip connections of the reaction module. The ADC module is the source of data monitored from external sensors, usually voltage information, which is used by the reaction channel modulation to generate a PWM signal control signal in order to maintain the load current within a certain predefined boundary. eTPU is commonly used as the source for controlling the activation of a reaction channel. Alternatively eMIOS or PIT can be used to provide that control signal.

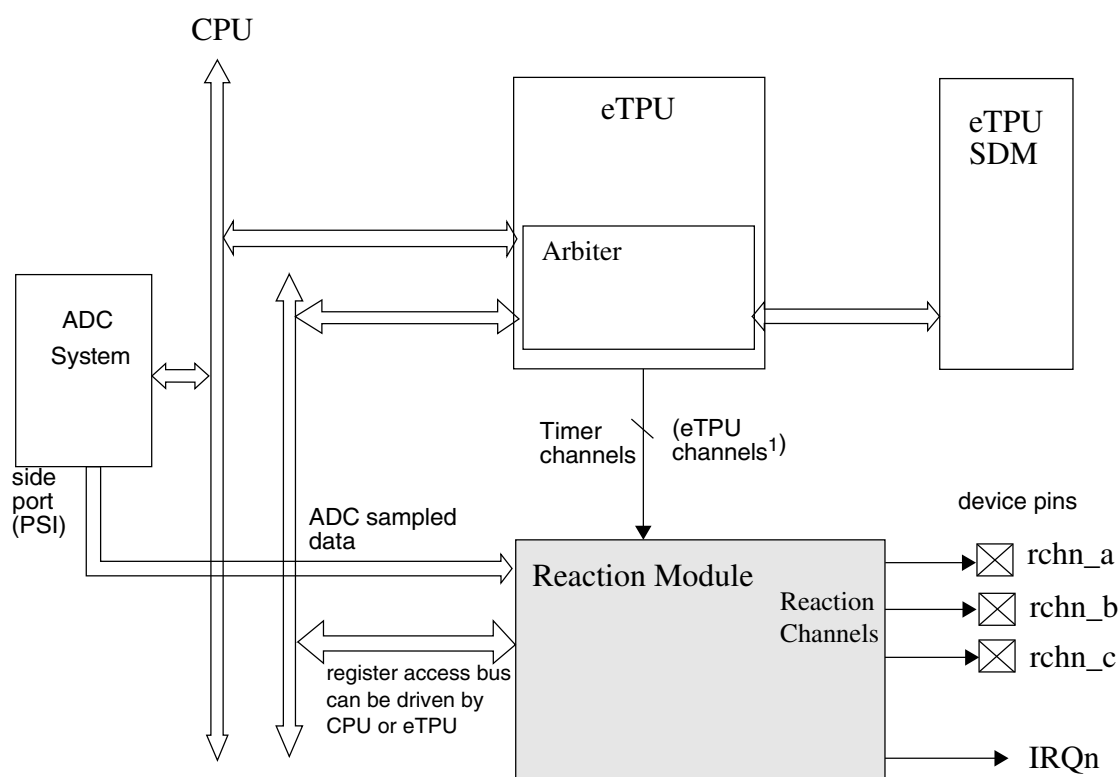


Figure 38-2. Reaction Module System Interconnection

The Reaction Module provides 10 channels with three outputs per channel. The channel outputs are programmed independently from each other. The outputs do not have a predefined ON or OFF state meaning as the definition of ON and OFF depends upon the application.

Figure 38-3 describes the interconnection between the reaction channel and the blocks in the Reaction Module. The indexes next to the blocks indicate the order in which each submodule executes an action during the modulation process. A simplified sequence of these steps is described as follows:

1. A new ADC result from the eQADC module is received, triggering one reaction channel
2. The triggered reaction channel generates the Modulation Word address based on the data programmed in the internal configuration register
3. A Modulation word is selected by the reaction channel
4. The selected Modulation word generates the address for the Threshold Bank
5. The Threshold bank selected data is compared against the incoming ADC result
6. The comparison result is sent back to the channel
7. The reaction channel executes modulation action and updates the channel output

In order to execute steps 1 through 7 the Reaction module takes five Module clock cycles.

The reaction channel sets the outputs according to HOD or LOD fields provided by the selected Modulation word.

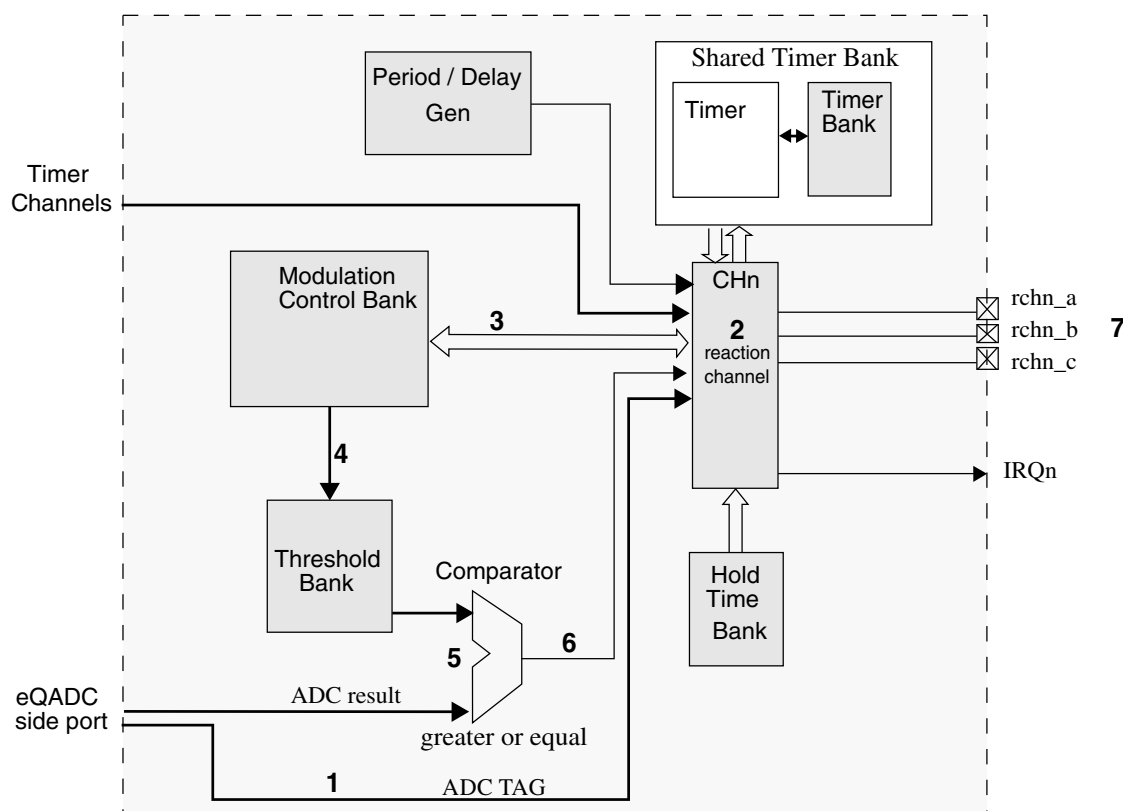


Figure 38-3. Channel Interaction with internal submodules

Figure 38-4 shows the Reaction Module's internal architecture. The channels are controlled by a Modulation Control Word provided by the Modulation Control Bank submodule. The Modulation Word has information about the type of modulation to be executed as well as the addresses for the Threshold and Timer banks. The channel stores information such as the address of the modulation word and initial values for the output pins after the channel is *enabled*.

Note that the modulation process only initiates after the channel is moved to the *active* state. This process will be described in more detail in the following sections.

After the activation, the channel sets the outputs with a predefined value stored in the channel configuration register. Only after this initialization process the channel is ready to start a modulation process. Note that the channel only considers an ADC result as valid if it is within a window defined by the timer input, usually connected to the on-chip eTPU. The Reaction Channel can access more than one Modulation Word by incrementing the internal MODULATION_ADDR field in the channel configuration register. The original value of this field is preserved thus when a new modulation cycle is initiated the same modulation address value can be used.

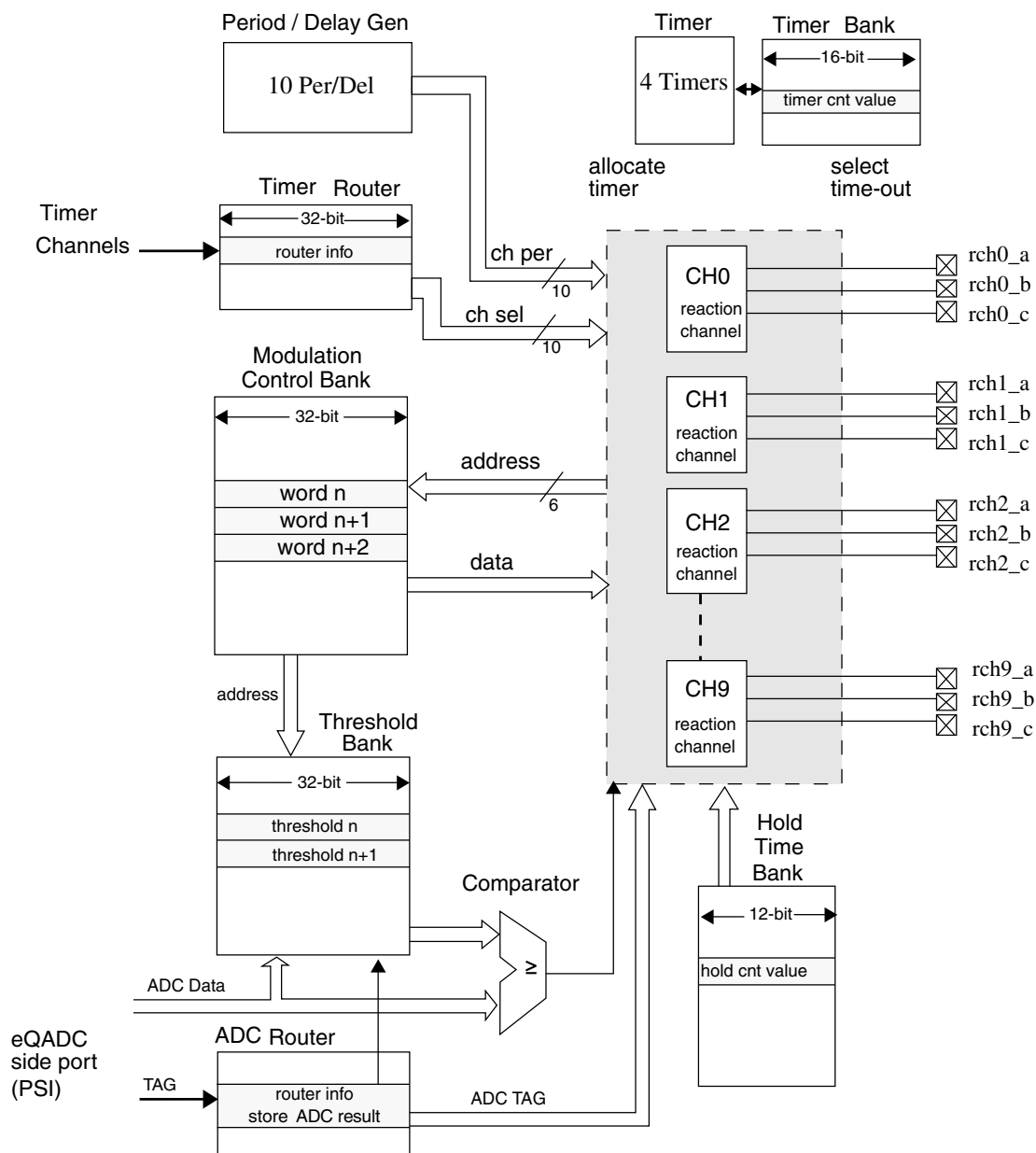


Figure 38-4. Reaction Module Block Diagram

Timer channels, such as eTPU channel outputs, are connected to the Timer Channel Router. This submodule routes each timer channel to a reaction channel. Note that one timer channel can be routed to more than one reaction channel.

The modulation process initiates when an ADC result arrives and the time window is active. The ADC Router sends a trigger signal to all channels indicating that an ADC result is available. The channel address resolution is based on the TAG field received with the incoming ADC result. After decoding the TAG field the channel accesses the Modulation Word using the information stored in the channel configuration register.

The Threshold Bank submodule stores values to be used on the comparison with incoming ADC results. The address for the stored values is generated by the Modulation Control Bank. This address generation is actually executed in a two-step process since the Modulation Word is first addressed by the channel which then generates the address for the Threshold Bank. After having both inputs defined the Comparator generates the comparison result back to the channels.

The Hold timer bank address is also stored in the modulation word. The reaction channel uses that information for the modulation process which requires the output to remain ON or OFF during a certain amount of time. The Hold counter itself is located inside each channel.

The Period Delay generator provides timing signals for the channels operating in the threshold-period modulation mode. It provides period starting signal for each channel with a configurable constant delay between two consecutive channels.

38.3 Signal Description

The following table shows the chip level signals for the Reaction module.

Table 38-2. Signal Properties

Name	Function
RCHn_a	Output pin 'a' of Reaction Channel 'n'
RCHn_b	Output pin 'b' of Reaction Channel 'n'
RCHn_c	Output pin 'c' of Reaction Channel 'n'

38.3.1 REACM2_RCHn — REACM Channel (n) Output Pin a, b and c

Each Reaction Channel provides three outputs to be connected to device primary output PADs. These outputs can be configured as necessary in order to meet application requirements. Thus the active state as well as overall functionality are defined by the registers within the Reaction Module and the defined sequence of operation. The reset state for these outputs is zero.

38.4 Memory Map and Register Definition

This section provides a detailed description of all Reaction Module registers accessible to the end user.

The REACM2 is in the eTPU address space to allow the eTPU direct access to the registers.

NOTE

For reset value of the REACM2_MCR, refer to the section describing this register.

REACM2 memory map

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
0	Module Configuration (REACM2_MCR)	32	R/W	0000_0000h	38.4.1/1852
4	Timer Configuration (REACM2_TCR)	32	R/W	0000_0000h	38.4.2/1854
8	Threshold Router Register (REACM2_THRR)	32	R/W	0000_0000h	38.4.3/1854
10	ADC Sensor Input Register (REACM2_SINR)	32	R/W	0000_0000h	38.4.4/1856
14	Period Generator Configuration Register (REACM2_PCR)	32	R/W	0000_0000h	38.4.5/1857
18	Period Shift Delay Configuration Register (REACM2_PSCR)	32	R/W	0000_0000h	38.4.6/1857
20	Global Error Flag Register (REACM2_GEFR)	32	R	0000_0000h	38.4.7/1858
100	Channel n Configuration Register (REACM2_CHCR0)	32	R/W	0000_0000h	38.4.8/1860
104	Channel n Status Register (REACM2_CHSR0)	32	R	0000_0000h	38.4.9/1863
108	Channel n Router Register (REACM2_CHRR0)	32	R/W	0000_0000h	38.4.10/1866
110	Channel n Configuration Register (REACM2_CHCR1)	32	R/W	0000_0000h	38.4.8/1860
114	Channel n Status Register (REACM2_CHSR1)	32	R	0000_0000h	38.4.9/1863
118	Channel n Router Register (REACM2_CHRR1)	32	R/W	0000_0000h	38.4.10/1866
120	Channel n Configuration Register (REACM2_CHCR2)	32	R/W	0000_0000h	38.4.8/1860
124	Channel n Status Register (REACM2_CHSR2)	32	R	0000_0000h	38.4.9/1863
128	Channel n Router Register (REACM2_CHRR2)	32	R/W	0000_0000h	38.4.10/1866
130	Channel n Configuration Register (REACM2_CHCR3)	32	R/W	0000_0000h	38.4.8/1860
134	Channel n Status Register (REACM2_CHSR3)	32	R	0000_0000h	38.4.9/1863
138	Channel n Router Register (REACM2_CHRR3)	32	R/W	0000_0000h	38.4.10/1866
140	Channel n Configuration Register (REACM2_CHCR4)	32	R/W	0000_0000h	38.4.8/1860
144	Channel n Status Register (REACM2_CHSR4)	32	R	0000_0000h	38.4.9/1863

Table continues on the next page...

REACM2 memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
148	Channel n Router Register (REACM2_CHRR4)	32	R/W	0000_0000h	38.4.10/1866
150	Channel n Configuration Register (REACM2_CHCR5)	32	R/W	0000_0000h	38.4.8/1860
154	Channel n Status Register (REACM2_CHSR5)	32	R	0000_0000h	38.4.9/1863
158	Channel n Router Register (REACM2_CHRR5)	32	R/W	0000_0000h	38.4.10/1866
160	Channel n Configuration Register (REACM2_CHCR6)	32	R/W	0000_0000h	38.4.8/1860
164	Channel n Status Register (REACM2_CHSR6)	32	R	0000_0000h	38.4.9/1863
168	Channel n Router Register (REACM2_CHRR6)	32	R/W	0000_0000h	38.4.10/1866
170	Channel n Configuration Register (REACM2_CHCR7)	32	R/W	0000_0000h	38.4.8/1860
174	Channel n Status Register (REACM2_CHSR7)	32	R	0000_0000h	38.4.9/1863
178	Channel n Router Register (REACM2_CHRR7)	32	R/W	0000_0000h	38.4.10/1866
180	Channel n Configuration Register (REACM2_CHCR8)	32	R/W	0000_0000h	38.4.8/1860
184	Channel n Status Register (REACM2_CHSR8)	32	R	0000_0000h	38.4.9/1863
188	Channel n Router Register (REACM2_CHRR8)	32	R/W	0000_0000h	38.4.10/1866
190	Channel n Configuration Register (REACM2_CHCR9)	32	R/W	0000_0000h	38.4.8/1860
194	Channel n Status Register (REACM2_CHSR9)	32	R	0000_0000h	38.4.9/1863
198	Channel n Router Register (REACM2_CHRR9)	32	R/W	0000_0000h	38.4.10/1866
300	Shared Timer Bank (REACM2_STBK0)	32	R/W	0000_0000h	38.4.11/1867
304	Shared Timer Bank (REACM2_STBK1)	32	R/W	0000_0000h	38.4.11/1867
308	Shared Timer Bank (REACM2_STBK2)	32	R/W	0000_0000h	38.4.11/1867
30C	Shared Timer Bank (REACM2_STBK3)	32	R/W	0000_0000h	38.4.11/1867
310	Shared Timer Bank (REACM2_STBK4)	32	R/W	0000_0000h	38.4.11/1867
380	Hold Timer Bank (REACM2_HOTBK0)	32	R/W	0000_0000h	38.4.12/1867
384	Hold Timer Bank (REACM2_HOTBK1)	32	R/W	0000_0000h	38.4.12/1867
388	Hold Timer Bank (REACM2_HOTBK2)	32	R/W	0000_0000h	38.4.12/1867
38C	Hold Timer Bank (REACM2_HOTBK3)	32	R/W	0000_0000h	38.4.12/1867
390	Hold Timer Bank (REACM2_HOTBK4)	32	R/W	0000_0000h	38.4.12/1867
400	Threshold Bank (REACM2_THBK0)	32	R/W	0000_0000h	38.4.13/1868

Table continues on the next page...

REACM2 memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
404	Threshold Bank (REACM2_THBK1)	32	R/W	0000_0000h	38.4.13/1868
408	Threshold Bank (REACM2_THBK2)	32	R/W	0000_0000h	38.4.13/1868
40C	Threshold Bank (REACM2_THBK3)	32	R/W	0000_0000h	38.4.13/1868
410	Threshold Bank (REACM2_THBK4)	32	R/W	0000_0000h	38.4.13/1868
414	Threshold Bank (REACM2_THBK5)	32	R/W	0000_0000h	38.4.13/1868
418	Threshold Bank (REACM2_THBK6)	32	R/W	0000_0000h	38.4.13/1868
41C	Threshold Bank (REACM2_THBK7)	32	R/W	0000_0000h	38.4.13/1868
420	Threshold Bank (REACM2_THBK8)	32	R/W	0000_0000h	38.4.13/1868
424	Threshold Bank (REACM2_THBK9)	32	R/W	0000_0000h	38.4.13/1868
428	Threshold Bank (REACM2_THBK10)	32	R/W	0000_0000h	38.4.13/1868
42C	Threshold Bank (REACM2_THBK11)	32	R/W	0000_0000h	38.4.13/1868
430	Threshold Bank (REACM2_THBK12)	32	R/W	0000_0000h	38.4.13/1868
434	Threshold Bank (REACM2_THBK13)	32	R/W	0000_0000h	38.4.13/1868
438	Threshold Bank (REACM2_THBK14)	32	R/W	0000_0000h	38.4.13/1868
43C	Threshold Bank (REACM2_THBK15)	32	R/W	0000_0000h	38.4.13/1868
440	Threshold Bank (REACM2_THBK16)	32	R/W	0000_0000h	38.4.13/1868
444	Threshold Bank (REACM2_THBK17)	32	R/W	0000_0000h	38.4.13/1868
448	Threshold Bank (REACM2_THBK18)	32	R/W	0000_0000h	38.4.13/1868
44C	Threshold Bank (REACM2_THBK19)	32	R/W	0000_0000h	38.4.13/1868
450	Threshold Bank (REACM2_THBK20)	32	R/W	0000_0000h	38.4.13/1868
454	Threshold Bank (REACM2_THBK21)	32	R/W	0000_0000h	38.4.13/1868
458	Threshold Bank (REACM2_THBK22)	32	R/W	0000_0000h	38.4.13/1868

Table continues on the next page...

REACM2 memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
45C	Threshold Bank (REACM2_THBK23)	32	R/W	0000_0000h	38.4.13/1868
460	Threshold Bank (REACM2_THBK24)	32	R/W	0000_0000h	38.4.13/1868
464	Threshold Bank (REACM2_THBK25)	32	R/W	0000_0000h	38.4.13/1868
468	Threshold Bank (REACM2_THBK26)	32	R/W	0000_0000h	38.4.13/1868
46C	Threshold Bank (REACM2_THBK27)	32	R/W	0000_0000h	38.4.13/1868
470	Threshold Bank (REACM2_THBK28)	32	R/W	0000_0000h	38.4.13/1868
474	Threshold Bank (REACM2_THBK29)	32	R/W	0000_0000h	38.4.13/1868
478	Threshold Bank (REACM2_THBK30)	32	R/W	0000_0000h	38.4.13/1868
47C	Threshold Bank (REACM2_THBK31)	32	R/W	0000_0000h	38.4.13/1868
600	ADC Result Maximum Limit Check (REACM2_ADCMAX)	32	R/W	0000_0000h	38.4.14/1869
680	Modulation Range Pulse Width (REACM2_RANGE_PWD)	32	R/W	0000_0000h	38.4.15/1869
6C0	Modulation Minimum Pulse Width (REACM2_MIN_PWD)	32	R/W	0000_0000h	38.4.16/1870
700	Modulation Control Word Bank (REACM2_MWBK0)	32	R/W	0000_0000h	38.4.17/1871
704	Modulation Control Word Bank (REACM2_MWBK1)	32	R/W	0000_0000h	38.4.17/1871
708	Modulation Control Word Bank (REACM2_MWBK2)	32	R/W	0000_0000h	38.4.17/1871
70C	Modulation Control Word Bank (REACM2_MWBK3)	32	R/W	0000_0000h	38.4.17/1871
710	Modulation Control Word Bank (REACM2_MWBK4)	32	R/W	0000_0000h	38.4.17/1871
714	Modulation Control Word Bank (REACM2_MWBK5)	32	R/W	0000_0000h	38.4.17/1871
718	Modulation Control Word Bank (REACM2_MWBK6)	32	R/W	0000_0000h	38.4.17/1871
71C	Modulation Control Word Bank (REACM2_MWBK7)	32	R/W	0000_0000h	38.4.17/1871
720	Modulation Control Word Bank (REACM2_MWBK8)	32	R/W	0000_0000h	38.4.17/1871
724	Modulation Control Word Bank (REACM2_MWBK9)	32	R/W	0000_0000h	38.4.17/1871

Table continues on the next page...

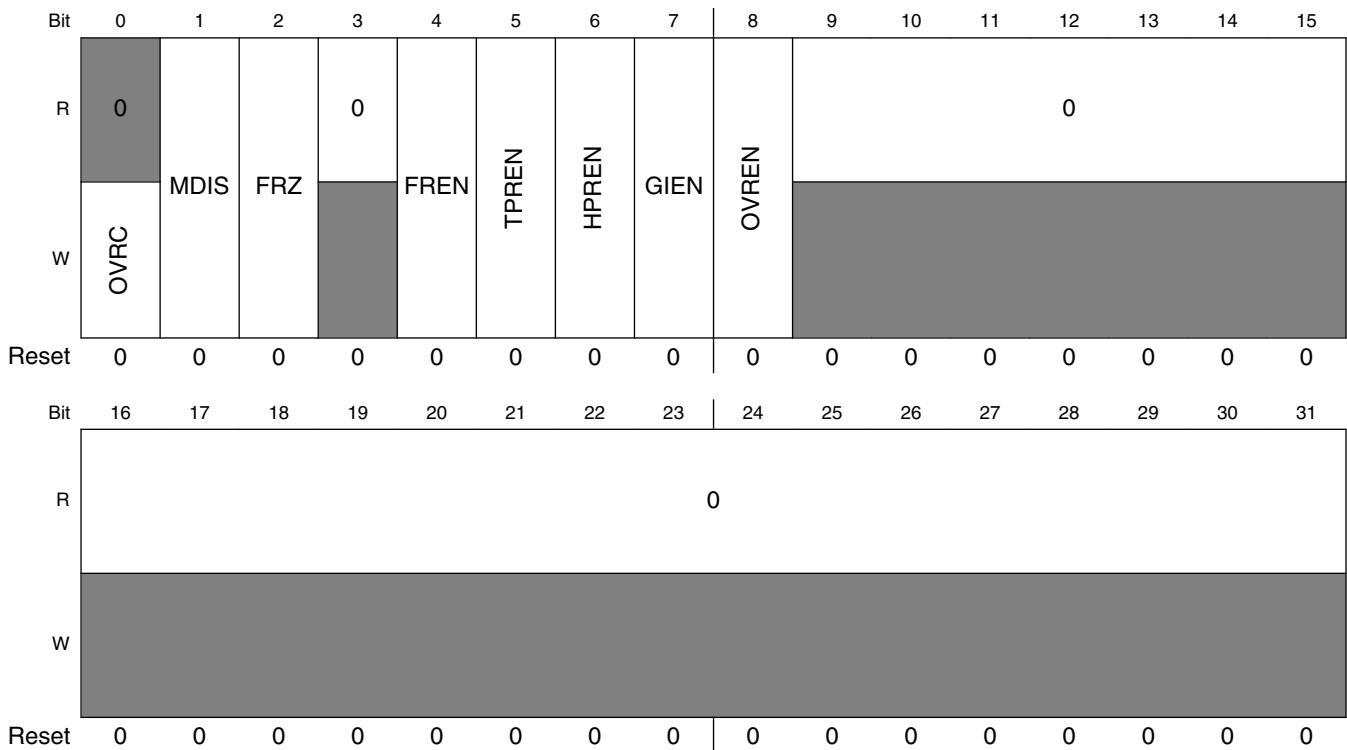
REACM2 memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
728	Modulation Control Word Bank (REACM2_MWBK10)	32	R/W	0000_0000h	38.4.17/1871
72C	Modulation Control Word Bank (REACM2_MWBK11)	32	R/W	0000_0000h	38.4.17/1871
730	Modulation Control Word Bank (REACM2_MWBK12)	32	R/W	0000_0000h	38.4.17/1871
734	Modulation Control Word Bank (REACM2_MWBK13)	32	R/W	0000_0000h	38.4.17/1871
738	Modulation Control Word Bank (REACM2_MWBK14)	32	R/W	0000_0000h	38.4.17/1871
73C	Modulation Control Word Bank (REACM2_MWBK15)	32	R/W	0000_0000h	38.4.17/1871
740	Modulation Control Word Bank (REACM2_MWBK16)	32	R/W	0000_0000h	38.4.17/1871
744	Modulation Control Word Bank (REACM2_MWBK17)	32	R/W	0000_0000h	38.4.17/1871
748	Modulation Control Word Bank (REACM2_MWBK18)	32	R/W	0000_0000h	38.4.17/1871
74C	Modulation Control Word Bank (REACM2_MWBK19)	32	R/W	0000_0000h	38.4.17/1871
750	Modulation Control Word Bank (REACM2_MWBK20)	32	R/W	0000_0000h	38.4.17/1871
754	Modulation Control Word Bank (REACM2_MWBK21)	32	R/W	0000_0000h	38.4.17/1871
758	Modulation Control Word Bank (REACM2_MWBK22)	32	R/W	0000_0000h	38.4.17/1871
75C	Modulation Control Word Bank (REACM2_MWBK23)	32	R/W	0000_0000h	38.4.17/1871
760	Modulation Control Word Bank (REACM2_MWBK24)	32	R/W	0000_0000h	38.4.17/1871
764	Modulation Control Word Bank (REACM2_MWBK25)	32	R/W	0000_0000h	38.4.17/1871
768	Modulation Control Word Bank (REACM2_MWBK26)	32	R/W	0000_0000h	38.4.17/1871
76C	Modulation Control Word Bank (REACM2_MWBK27)	32	R/W	0000_0000h	38.4.17/1871
770	Modulation Control Word Bank (REACM2_MWBK28)	32	R/W	0000_0000h	38.4.17/1871
774	Modulation Control Word Bank (REACM2_MWBK29)	32	R/W	0000_0000h	38.4.17/1871
778	Modulation Control Word Bank (REACM2_MWBK30)	32	R/W	0000_0000h	38.4.17/1871
77C	Modulation Control Word Bank (REACM2_MWBK31)	32	R/W	0000_0000h	38.4.17/1871

38.4.1 Module Configuration (REACM2_MCR)

The REACM2_MCR contains the control bits to configure the general operation of the Reaction Module.

Address: 0h base + 0h offset = 0h



REACM2_MCR field descriptions

Field	Description
0 OVRC	Overrun Detection Flag Clear The OVRC clears the OVR flag if "1" is written. This bit reads always as 0x0. If a set event occurs at the same time a flag clear is done, the set event has precedence over the clear thus the flag remains set. 0 No action 1 Clears OVR bit
1 MDIS	Module Disable The MDIS bit puts the Reaction Module in low power mode. Communication through the slave-bus Interface is ignored in this mode except writes to the REAC_MCR register which are allowed, except for the FRZ and FREN bits. The global debug signal state is not changed internally while in low power mode. 0 Normal Mode 1 Low Power Mode

Table continues on the next page...

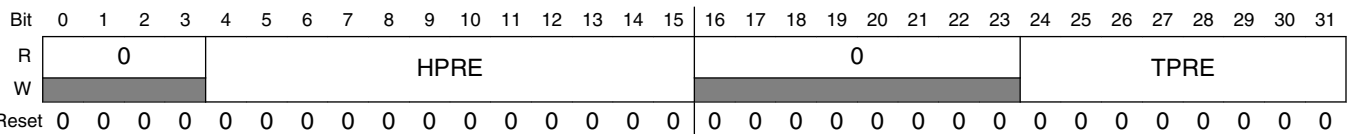
REACM2_MCR field descriptions (continued)

Field	Description
2 FRZ	Freeze Control The FRZ bit controls the state of the Reaction Module regarding debug operation. If both FREN and FRZ bits are set the module enters debug mode. In this mode all time bases are halted and the outputs of channels are controlled solely by software. See section Debug Mode. This bit cannot be written if MDIS bit is set. 0 Normal Mode 1 Debug Mode
3 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
4 FREN	Freeze Enable The FREN bit enables the Reaction Module to enter debug mode. The debug mode is controlled either by the FRZ bit or by a global debug signal. This bit cannot be written if MDIS bit is set. 0 Debug Mode disabled 1 Debug Mode enable
5 TPREN	Timer Prescaler Enable The TPREN bit enables the Shared Timer Prescaler in the Reaction Module. 0 Prescaler Disabled 1 Prescaler Enabled
6 HPREN	Hold Timer Prescaler Enable The HPREN bit enables the Hold Prescaler in the Reaction Module. 0 Prescaler Disabled 1 Prescaler Enabled
7 GIEN	Global Interrupt Enable The GIEN bit enables the assertion of the global interrupt request to the CPU when any of the channel interrupts is set or when the OVR flag and its corresponding enable OVREN are both set. Note that this signal does not control the individual channel interrupt request. See section Global Interrupt and Channel Interrupt for more details. 0 Interrupt disabled 1 Interrupt enabled
8 OVREN	Overflow Detection Interrupt Enable The OVREN enables the OVR flag, when set, to generate a global interrupt request for the CPU. 0 Interrupt disabled 1 Interrupt enabled
9–31 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

38.4.2 Timer Configuration (REACM2_TCR)

The REACM2_TCR Timer Configuration Register contains the prescaler settings to configure the operation of the Reaction Module Hold Timer and Shared Timers. It is mandatory to change the value of the prescalers either when the prescalers are disabled by its control enable bits HPREN and TPREN in the Configuration Register or when the counters are not being used by any channel. Note that the prescalers are completely independent and thus modifying HPRE does not affect TPRES and vice-versa.

Address: 0h base + 4h offset = 4h



REACM2_TCR field descriptions

Field	Description
0–3 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
4–15 HPRE	Hold Timer Prescaler The HPRE field defines the rate of the Hold Timers on each Reaction Channel. If its value is zero the prescaler is bypassed thus the Hold timer operates at the module clock rate. If HPRE = 0x01 the module operates at module clock divide by two and so forth up to HPRE = 0xFFFF which defines system clock divided by 4096.
16–23 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
24–31 TPRE	Timer Prescaler The TPRES field defines the rate of the Timers on the Timer bank. If its value is zero the prescaler is bypassed thus the Timer operates at the module clock rate. If TPRES = 0x01 the Timer operates at module clock divide by two and so forth up to TPRES = 0xFF which defines module clock divided by 256 as the frequency of operation for the Timer.

38.4.3 Threshold Router Register (REACM2_THRR)

The Reaction Module Threshold Router Register routes the ADC result to the Threshold Bank. Since a TAG is assigned for each ADC result coming from the on-chip ADC module, this TAG is specified in this register and used for the routing process. The ADC result is written to the Threshold Value Bank as soon as it is received by the Reaction Module. Note that THRADC0 and THRADC1 TAG values may also be used by the Reaction Channels. In this case the results are routed to both, the channel and the Threshold Bank.

NOTE

Due to the timing in which these parallel events occur, if the channel uses the same Threshold Bank address in which the incoming ADC result was written to, this ADC result is used for the channel modulation being executed.

Address: 0h base + 8h offset = 8h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0						WREN1	WREN0	0							
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31				
R	0				THRADC1								0				THRADC0			
W																				
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0				

REACM2_THRR field descriptions

Field	Description
0–5 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
6 WREN1	Write Enable Bit for THRADC1 The WREN1 write enable bit 1 controls if the ADC result having a TAG matching to THRADC1 field will be written into address one of the Threshold Bank 0 Do not write received ADC result to Threshold bank 1 Write received ADC result to Threshold bank address one
7 WREN0	Write Enable Bit for THRADC0 The WREN0 write enable bit 0 controls if the ADC result having a TAG matching to THRADC0 field will be written into address zero of the Threshold Bank 0 Do not write received ADC result to Threshold bank 1 Write received ADC result to Threshold bank address zero
8–19 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
20–23 THRADC1	ADC result Router value for Threshold Bank address one The THRADC1[3:0] field controls the routing from the received ADC result to the Threshold Bank address one. Any ADC result which TAG matching THRADC1 will be routed to Threshold Bank.
24–27 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
28–31 THRADC0	ADC result Router for Threshold Bank address zero

Table continues on the next page...

REACM2_THRR field descriptions (continued)

Field	Description
	The THRADC0[3:0] field controls the routing from the received ADC result to the Threshold Bank address zero. Any ADC result which TAG matching THRADC0 will be routed to Threshold Bank.

38.4.4 ADC Sensor Input Register (REACM2_SINR)

The Reaction Module ADC Sensor Input Register is used to monitor the ADC Interface (see section [ADC Interface](#)) and allows the software to define the ADC result and TAG values for the Reaction Module. Also ADC values captured through DMA directly from the ADC interface or after filtering can be transferred to the Reaction Module by the CPU. Writes to this register may overwrite data coming from the ADC interface thus it should be used when no ADC conversion has the Reaction Module as the target. Writing to this register triggers the Reaction Channel selected by the TAG value to execute a comparison and to evaluate an eventually new value for the channel outputs.

Address: 0h base + 10h offset = 10h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0												ADC_TAG				ADC_RESULT															
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	

REACM2_SINR field descriptions

Field	Description
0–11 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
12–15 ADC_TAG	TAG value The ADC_TAG[3:0] represents the TAG for the ADC conversion which is used by the Reaction Module to select the Reaction Channel to execute the modulation.
16–31 ADC_RESULT	ADC conversion result value The ADC_RESULT represents the value resulting from an ADC conversion. This value is used for the Reaction Channel Modulation process or for capturing by the Threshold Bank.

38.4.5 Period Generator Configuration Register (REACM2_PCR)

The Reaction Module Period Generator Configuration Register is used to setup the period pulse generator sub-block. See section [Period / Delay Generator](#). This configuration is necessary when running the modulation mode threshold/period.

Address: 0h base + 14h offset = 14h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0								PERPRESC								PER															
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	

REACM2_PCR field descriptions

Field	Description
0–7 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
8–15 PERPRESC	Period prescaler rate value The PERPRESC field defines the rate of the prescaler to be used by the Period counter and Delay counter. If its value is 0, the rate is 1, and so forth up to PERPRESC= 0xFF which defines to divide the system clock by 256.
16–31 PER	Period time value The PER represents the value for the internal period generator counter. When it is configured to all 0, the counter is stopped. And if not all 0, the period time in clock cycles is given by multiplying (PER+1) by the prescaler value (PERPRESC+1).

38.4.6 Period Shift Delay Configuration Register (REACM2_PSCR)

The Reaction Module Period Shift Delay Configuration Register is used to configure the delay time between period pulses of two consecutive channels. See section [Period / Delay Generator](#). The delay multiplied by the number of channels must be less than or equal to the period.

Address: 0h base + 18h offset = 18h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0															DLY																
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	

REACM2_PSCR field descriptions

Field	Description
0–15 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
16–31 DLY	Delay value between two consecutive channel period pulses The DLY represents the value to configure the shift delay between the period pulses of two consecutive reaction channels (for example, CH0 and CH1). The correct time depends on the prescaler counter. Also, if DLY=0 then the period pulses of all reaction channels occur at the same time. If DLY is not all 0, then the shift delay time in clock cycles is given by multiplying DLY with the prescaler (PERPRESC+1).

38.4.7 Global Error Flag Register (REACM2_GEFR)

The Reaction Module Global Error Flag Register helps the software in the resolution of error conditions signaled by the Reaction Channels. This allows a faster service in error handling. This register mirrors the Error Flag in the Channel Configuration and Status registers.

Address: 0h base + 20h offset = 20h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	OVR	0														
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0						EF9	EF8	EF7	EF6	EF5	EF4	EF3	EF2	EF1	EF0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

REACM2_GEFR field descriptions

Field	Description
0 OVR	Overflow Detection Flag The OVR flag is used to indicate that an overflow condition was detected at the ADC Interface. See section ADC Interface. 0 Overflow not detected 1 Overflow detected
1–21 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
22 EF9	Error Flag The EFn error flag bit indicates an error condition occurred in channel n. This bit is provided for a fast channel error resolution in the Reaction module. The condition could be any error indicated by MAXL,

Table continues on the next page...

REACM2_GEFR field descriptions (continued)

Field	Description
	OCDF, SCDF, SQER, RAER or TAER error flags as described in section "REACM2 Channel Status Register (REACM2_CHSR)". The EFn bit is automatically cleared if the corresponding channel flags are all cleared.
23 EF8	Error Flag See description for EF9.
24 EF7	Error Flag See description for EF9.
25 EF6	Error Flag See description for EF9.
26 EF5	Error Flag See description for EF9.
27 EF4	Error Flag See description for EF9.
28 EF3	Error Flag See description for EF9.
29 EF2	Error Flag See description for EF9.
30 EF1	Error Flag See description for EF9.
31 EF0	Error Flag See description for EF9.

38.4.8 Channel n Configuration Register (REACM2_CHCRn)

The Reaction Module Channel Configuration Register controls the channel behavior.

Address: 0h base + 100h offset + (16d × i), where i=0d to 9d

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	CHEN		SWMC	MAXLEN	OCDFEN	SCDFEN	TAEREN	SQEREN	RAEREN	0	CHOFF	SDCM	0	DOFF		
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0				BSB				0				MODULATION_ADDR			
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

REACM2_CHCRn field descriptions

Field	Description
0–1 CHEN	Channel Enable Bits The CHEN[1:0] bits control the activation of the Reaction Channel. Once changed from disabled, CHEN = 00, to any enabled state, the channel reads the first Modulation Word and initializes itself in order to be ready to execute a modulation as soon as a timer control window is detected or SWMC=1, if software modulation control is selected by CHEN= 11. NOTE: If CHEN is programed with 01 or 11 enabling a channel, and the eTPU time window is already active (or SWMC=1), the Reaction Channel disregards this window and waits until the next window activation in order to start the modulation process by moving to the active state. In order to start a modulation controlled by software, it is required to first write CHEN=11 and then to write SWMC=1 only after that. 00 Channel disabled, meaning that it does not execute any modulation even if a timer window is detected or SWMC is made active. In this mode DOFF field in the Channel Configuration register defines the state of the channel outputs. 01 Channel is enabled for timer control only, meaning that as soon as a timer window is detected a modulation sequence starts. 10 Reserved 11 Channel enabled for software control only, meaning that as soon as SWMC bit is set a modulation sequence starts.
2 SWMC	Software Modulation Control bit The SWMC, Software Modulation Control bit, if set, initiates the channel modulation. It is equivalent to the assertion of an eTPU channel connected to the Reaction module. To use this functionality it is required that CHEN[1:0] = 11.

Table continues on the next page...

REACM2_CHCRn field descriptions (continued)

Field	Description
	0 Channel does not perform modulation 1 Channel executes modulation
3 MAXLEN	ADC result Maximum Limit Interrupt Enable bit The MAXLEN enables the MAXL flag, when set, to generate an interrupt request for the CPU. 0 Interrupt disabled 1 Interrupt enabled
4 OCDFEN	OCDF Interrupt Enable bit The OCDFEN bit enables the OCDF bit to issue an interrupt request. 0 OCDF interrupt disabled 1 OCDF interrupt enabled
5 SCDFEN	SCDF Interrupt Enable bit The SCDFEN bit enables the SCDF bit to issue an interrupt request. 0 SCDF interrupt disabled 1 SCDF interrupt enabled
6 TAEREN	TAER Interrupt Enable bit The TAEREN bit enables the TAER bit to issue an interrupt request. 0 TAER interrupt disabled 1 TAER interrupt enabled
7 SQEREN	Modulation Word Sequence Error Interrupt Enable The SQEREN bit enables the SQER flag to generate an interrupt request. 0 SQER interrupt disabled 1 SQER interrupt enabled
8 RAEREN	Resource Allocation Error Interrupt Enable The RAEREN bit enables the RAER flag to generate an interrupt request. 0 RAER interrupt disabled 1 RAER interrupt enabled
9 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
10 CHOFF	Output Disable bit The CHOFF bit allows disabling the output pin. DOFF[2:0] field is used as disable state for the channel outputs. The output pins will set to DOFF value if CHOFF is asserted. Note that disabling the channel through CHOFF does not disable the channel operation. Only the channel outputs are forced to DOFF state. 0 Output Disable Disabled 1 Output Disable Enabled
11 SDCM	Short Detection Counting Mode

Table continues on the next page...

REACM2_CHCR_n field descriptions (continued)

Field	Description
	The SDCM bit configures the short detector to count one or two short detection events to set the SCDF field. 0 SCDF is set if two short detection events occur in a row 1 SCDF is set if a short detection occurs
12 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
13–15 DOFF	Drive Off Control field The DOFF[2:0] field defines the Reaction Channel output disabled state. This condition is achieved either when CHEN= 00, or after CHEN activation, or when Output is disabled by CHOFF in the Channel Configuration register. • DOFF[0] bit defines CH_n_a output pin value • DOFF[1] bit defines CH_n_b output pin value • DOFF[2] bit defines CH_n_c output pin value NOTE: It is possible to control the channel outputs directly through the software by writing to the DOFF bits. In this case the channel should be in the disabled state, CHEN=00. NOTE: If the DOFF value is changed just after the channel is enabled, it is not assured the new DOFF value is immediately used in the channel output.
16–20 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
21–23 BSB	Bank Support Bits The BSB[2:0] provides control for a banked mode operation of the Reaction Module. Each bit in this field controls the channel outputs chn_c, chn_b and chn_a respectively. When asserted the channel output implements a sharing function with the corresponding output of the subsequent channel.
24–26 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
27–31 MODULATION_ ADDR	Address for Modulation Control Bank The MODULATION_ADDR field has the address of the Modulation Word in the Modulation Control Bank. This address is used as a base address for the first modulation word. The Reaction Channel can access subsequent words in the Modulation Control Bank by incrementing the Modulation Address field. Note that this field is not modified by the Reaction Channel in this process. NOTE: The number of subsequent words is limited to 15. Therefore the channel can handle a sequence with as many as 16 modulation words. NOTE: If a sequence advance is requested in the 16th modulation word, the channel returns back and accesses the first modulation word that is indicated by the address field MODULATION_ADDR on the REACM2_CHCR_n register.

38.4.9 Channel n Status Register (REACM2_CHSRn)

The Reaction Module Channel Status Register provides access to the channel flags and flag clear bits. It also provides access to the current values of the channel output being driven and the Modulation Word being accessed by the channel.

Address: 0h base + 104h offset + (16d × i), where i=0d to 9d

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0		MODACT	MAXL	OCDF	SCDF	TAER	SQER	RAER	CHOUT			0			
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0			0	0	0	0	0	0	0	MODULATION_POINTER					
W				MAXLC	OCDFC	SCDFC	TAERC	SQERC	RAERC							
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

REACM2_CHSRn field descriptions

Field	Description
0–1 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
2 MODACT	Modulation Active Flag The MODACT flag indicates that the channel is enabled and active thus executing a modulation cycle. 0 Channel is not active 1 Channel is active

Table continues on the next page...

REACM2_CHSR_n field descriptions (continued)

Field	Description
3 MAXL	ADC Maximum Limit Detection Flag The MAXL flag indicates that the ADC result which TAG is addressing this channel achieved or passed the maximum allowed limit specified in the ADCMAX register. This flag is set when the channel is in the active state (see section Channel Modes) or in debug mode. This bit is not set in the off time of the Threshold/Hold-off modulation mode. 0 Normal operation 1 ADC Maximum Limit Detected
4 OCDF	Open Circuit Detection Flag The OCDF flag indicates that an open circuit was detected in the channel load using the PWM monitored modulation, as described in Monitored Modulation. This Flag is set only if the channel is in the active state. 0 Normal operation 1 Open Circuit Detected
5 SCDF	Short Circuit Detection Flag The SCDF flag indicates that a short circuit was detected in the channel load using the PWM monitored modulation, as described in Monitored Modulation. This flag is set only if the channels are in the active state. 0 Normal Operation 1 Short Circuit Detected
6 TAER	Timer Allocation Error The Timer Allocation Error bit indicates that the channel tried to allocate a timer counter in the Shared Timer Bank without success. This situation is an indication that the Timer resources available in the module are not sufficient to execute the required functionality. This error indication is used during software development and should not occur during normal use of the module since it may result in incorrect operation. This flag is set only if the channel is in the active state. 0 No error occurred during timer allocation. 1 Error occurred during timer allocation
7 SQER	Modulation Word Sequence Error Flag The SQER bit indicates a Modulation Sequence error occurred, meaning that the time window which defines a modulation cycle ended in a premature modulation phase. The correct modulation phase for the time window to close is when SM field in the Modulation Word indicates no advance, or SM= 00. This Flag is set only if the channel is in the active state. 0 Modulation Sequence Error did not occur 1 Modulation Sequence Error occurred
8 RAER	Resource Allocation Error Flag The RAER bit indicates that a resource allocation error occurred. The possible allocation errors are: Modulation Control Word address is out of available range (including if the MODULATION_POINTER increments to an inexistent MCW address or wraps to 0x0), Threshold Value Bank address is out of available range, Hold Timer Bank address is out of available range, Shared Timer Bank address is out of available range, Channel Input Router points to an inexistent eTPU channel, and when the Hold timer is select for both modulation and sequence advance at the same time (i.e, SM=10 and MM=01). Note that the amount of hardware resources is configuration dependent thus may vary according to module

Table continues on the next page...

REACM2_CHSRn field descriptions (continued)

Field	Description
	parameterization in the SoC. This flag is intended to help on the debug of the Reaction Module during software development and can be set only if the channel is in the enabled state. NOTE: The condition that sets the RAER bit must be resolved prior to clear the bit, otherwise the bit can be set again. 0 No allocation error occurred 1 Allocation error occurred
9–11 CHOUT	Channel Output Monitoring bits The CHOUT[2:0] Channel Output Monitoring bits provides for the software the ability to monitor the output provided by the channel. This data is not buffered thus represents the channel output at the time the CPU read access is done. CHOUT[0] corresponds to the chn_a output, CHOUT[1] corresponds to the chn_b output and CHOUT[2] corresponds to the chn_c output pin. These bits are available independent of the channel mode or state.
12–18 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
19 MAXLC	ADC Maximum Limit Flag Clear The MAXLC clears the MAXL flag if "1" is written. This bit reads always as 0x0. If a set event occurs at the same time a flag clear is done, the set event has precedence over the clear thus the flag remains set. 0 No action 1 Clears MAXL bit
20 OCDFC	Open Circuit Detection Flag Clear The OCDFC clears the OCDF flag if "1" is written. This bit reads always as 0x0. If a set event occurs at the same time a flag clear is done, the set event has precedence over the clear thus the flag remains set. 0 No action 1 Clears OCDF bit
21 SCDFC	Short Circuit Detection Flag clear The SCDFC bit clears the SCDF flag if "1" is written. This bit is self negated thus read always as '0'. If a set event occurs at the same time a flag clear is done, the set event has precedence over the clear thus the flag remains set. 0 No action 1 Clears SCDF
22 TAERC	TAER Clear bit The TAERC bit clears the TAER bit if "1" is written. This bit is self-negated thus reads always as '0'. If a set event occurs at the same time a flag clear is done, the set event has precedence over the clear thus the flag remains set. 0 No action 1 Clears TAER
23 SQERC	Modulation Word Sequence Error Flag clear The SQERC bit clears the SQER flag in the Channel Status register. 0 SQER flag is not cleared 1 SQER flag is cleared

Table continues on the next page...

REACM2_CHSR_n field descriptions (continued)

Field	Description
24 RAERC	Resource Allocation Error Flag clear The RAERC bit clears the RAER flag in the Channel Status register. 0 RAER flag is not cleared 1 RAER flag is cleared
25 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
26–31 MODULATION_ POINTER	Modulation bank address generated by the channel The MODULATION_POINTER[5:0] gives to the software the information of the current address being generated by the channel to access the Modulation Word. Note that the address increments from a base address stored in the channel, MODULATION_ADDR[5:0]. This register is not buffered thus represents the current address being generated.

38.4.10 Channel n Router Register (REACM2_CHRR_n)

The Reaction Module Channel Router Register controls the channel connection to external timers, such as eTPU and eMIOS and ADC result data. The channels have access to any 16 timer inputs and any ADC sample tag from 0 to 15. Note that this architecture allows several Reaction Channels to point to the same timer channel or to the same ADC result.

Address: 0h base + 108h offset + (16d × i), where i=0d to 9d

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0												ADCR				0												CHIR			
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	

REACM2_CHRR_n field descriptions

Field	Description
0–11 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
12–15 ADCR	ADC result router field The ADCR[3:0] field selects which ADC result is used by the Reaction Channel for the modulation. The TAG[3:0] received along with the ADC result is used for comparison with the ADCR[3:0] field in order to define if the received result is used by the Reaction Channel.
16–27 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
28–31 CHIR	Channel Input router field The CHIR field selects which eTPU channel is used by the Reaction Channel for the modulation.

38.4.11 Shared Timer Bank (REACM2_STBKn)

The Reaction Modulation Shared Timer Bank is a set of registers which define the values used by the Reaction Module Timer. The timer values are programmed by the Host CPU during the configuration of the Reaction Module. Modulation Word accessed by the Reaction Channel contains the address of a specific Timer value stored in the Shared Timer bank. By selecting a Timer value the Reaction Channel also selects and enables a counter. When this counter reaches the selected Timer value a time-out indication is generated for the Reaction Channel that initiated the counter. This event is used for example to indicate that a next Modulation Word should be used for the modulation.

Address: 0h base + 300h offset + (4d × i), where i=0d to 4d

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0																SHARED_TIMER															
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	

REACM2_STBKn field descriptions

Field	Description
0–15 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
16–31 SHARED_TIMER	Timer Value The SHARED_TIMER value is one element of the Timer Register Bank. Up to 5 timer values can be stored within the Timer Bank. NOTE: When using the shared timer for sequence advance, the counted time (considering prescaler) must be greater than 64 clock cycles.

38.4.12 Hold Timer Bank (REACM2_HOTBKn)

The Reaction Module Hold Timer Bank is a set of registers that defines the values used by the Reaction Channels to measure hold time on certain modulation schemes. The timer values are programmed by software and addressed by the Reaction Channel based on the data read from a Modulation Word.

Address: 0h base + 380h offset + (4d × i), where i=0d to 4d

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0																HOLD_TIM															
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	

REACM2_HOTBK n field descriptions

Field	Description
0–19 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
20–31 HOLD_TIM	Hold Timer Value The HOLD_TIM value is one element of the Hold Timer Register Bank. Up to 5 timer values can be stored within the Hold Timer Bank. NOTE: When using the hold timer for sequence advance, the counted time (considering prescaler) must be greater than 64 clock cycles.

38.4.13 Threshold Bank (REACM2_THBK n)

The Reaction Module Threshold Bank holds the value to be used for comparison against results received from the ADC. Based on that comparison the Reaction Channel decides the Channel output value to be either HOD or LOD.

Address: 0h base + 400h offset + (4d × i), where i=0d to 31d

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0																THRESHOLD_VALUE															
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	

REACM2_THBK n field descriptions

Field	Description
0–15 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
16–31 THRESHOLD_VALUE	Threshold Value The THRESHOLD_VALUE unsigned value is one element of the Threshold register used for a threshold modulation. Up to 32 timer values can be stored within the Threshold Bank.

38.4.14 ADC Result Maximum Limit Check (REACM2_ADCMAX)

The Reaction Module ADC Maximum Limit Check register holds the maximum expected value of the ADC result. If a value greater than or equal to ADC_MAX_LIMIT is received the MAXL bit in the corresponding Channel Status Register Error is asserted.

Address: 0h base + 600h offset = 600h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0																ADC_MAX_LIMIT															
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	

REACM2_ADCMAX field descriptions

Field	Description
0–15 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
16–31 ADC_MAX_LIMIT	Maximum Limit allowed for the ADC result The ADC_MAX_LIMIT[15:0] value indicates the maximum expected value for the ADC result. The MAXL bit in the corresponding channel Status register is set in case a greater or equal value is received from the ADC. If set to zero no limit checking is performed, thus MAXL bit will not be set anyway. NOTE: ADC results are always considered unsigned unless specific note states the contrary. Since ADC received results can be 14-bit values a two bit sign extension must be performed before any comparison is executed.

38.4.15 Modulation Range Pulse Width (REACM2_RANGE_PWD)

The Reaction Module Range PulseWidth Detection Check register provides the value used to check if the PWM pulse width generated during the modulation process is larger than a maximum pulse width specified by (MIN_PWD + RANGE_PWD). The checking is performed by the channel logic during appropriate time intervals. See section [Monitored Modulation](#). This function is implemented by sharing the use of Reaction Channel Hold counter, thus if the Hold timer is used by the channel sequence mode, SM in the Channel Configuration register, this checking function is not active.

Address: 0h base + 680h offset = 680h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0																RANGE_PWD															
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	

REACM2_RANGE_PWD field descriptions

Field	Description
0–19 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
20–31 RANGE_PWD	Range Pulse Width The RANGE_PWD[11:0] field defines the maximum pulse width allowed by the channel PWM generation. The checking is performed by the channel internal Hold Timer at appropriate times in the PWM modulation process. The maximum pulse is defined as (RANGE_PWD + MIN_PWD). See “REACM2 Minimum Pulse Width Check (REACM2_MIN_PWD)” for MIN_PWD description. If RANGE_PWD = 0x00 then no maximum pulse width checking is performed. NOTE: The RANGE_PWD value should be calculated considering the prescaler settings used for the HOLD counter. For a programmed (RANGE_PWD + MIN_PWD) value, a pulse narrower than or equal to (RANGE_PWD + MIN_PWD + 1) does not set the OCDF flag.

38.4.16 Modulation Minimum Pulse Width (REACM2_MIN_PWD)

The Reaction Module Minimum PulseWidth Check register provides the value used to check if the pulse width generated during the PWM modulation process on the channel outputs is shorter than a minimum pulse width specified by this register. The checking is performed by using the channel internal Hold timer during appropriate times on the channel operation when this counter is not being used for the hold modulation cycle.

Address: 0h base + 6C0h offset = 6C0h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R																																
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

REACM2_MIN_PWD field descriptions

Field	Description
0–19 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
20–31 MIN_PWD	Minimum Pulse Width The MIN_PWD[11:0] field defines the minimum pulse width allowed by the channel PWM generation. The checking is performed by the channel hold timer. If MIN_PWD[11:0] = 0x00 then no checking is done even for maximum pulse width or minimum pulse width. NOTE: The MIN_PWD value should be calculated considering the prescaler settings used for the HOLD counter. For a programmed MIN_PWD value, a pulse wider than (MIN_PWD+1) does not set the SCDF flag.

38.4.17 Modulation Control Word Bank (REACM2_MWBK n)

The Reaction Module Control Word Bank is a set of registers that controls the Reaction Channel Modulation scheme. These registers are programmed by software and read by the Reaction Channels. All the information required by the Reaction Channels to perform a modulation is stored in these words. All channels have access to the same words thus sharing of Modulation Words among channels is possible in this architecture.

There are 32 modulation control words in this device.

Address: 0h base + 700h offset + (4d × i), where i=0d to 31d

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R			0			0			0				0			
W	LOOP	IOSS		MM			SM			HOD					LOD	
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0							0					0			
W					THRESPT					STPT						HDOFFTPT
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

REACM2_MWBK n field descriptions

Field	Description
0 LOOP	Loop Control Bit The LOOP Control bit indicates that the next Modulation Control Word accessed by the Reaction Channel when a Modulation Word address increment event occurs should be the one indicated by the Channel MODULATION_ADDR[5:0] register with no offset. 0 Increment to the next modulation word 1 Loop back to initial channel modulation word
1 IOSS	Initial Output State Selection The IOSS bit defines the state of the channel output pins when a modulation phase is started by the Reaction Channel. This bit is important in the Modulation Word first access. This first access occurs after the channel is enabled by CHEN bit field in the Channel Configuration register, or when the modulation phase advances, e.g. due to a timer event. This bit also defines the modulation initial behavior for some modulation modes. See Modulation Mode MM[1:0] field for more details. 0 LOD[2:0] is initially used for chn_c, chn_b and chn_a respectively 1 HOD[2:0] is initially used for chn_c, chn_b and chn_a respectively
2 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

Table continues on the next page...

REACM2_MWBK n field descriptions (continued)

Field	Description
3–4 MM	Modulation Mode The MM[1:0] Modulation Mode field indicates the type of the modulation that is executed by the channel while the IOSS bit determines the modulation mode initial state. The initial behavior of the modulation modes (or the threshold comparison level performed in threshold/Period modulation) is selected by the IOSS bit. 00 Modulation Type: threshold/threshold Modulation occurs between two threshold levels defined by THRESPT upper level threshold pointer and THRESPT +1 which corresponds to a lower level threshold pointer. IOSS selects the initial output state. Output ON state is defined by HOD (IOSS=1) and Output OFF state is defined by LOD (IOSS=0). 01 Modulation Type: threshold/Hold-off - IOSS="0": Modulation starts with channel outputs set to LOD for an amount of time defined by the hold timer. After that outputs are set to HOD until an upper threshold defined by THRESHPT is reached. - IOSS="1": Modulation starts with channel outputs set to HOD until an upper threshold defined by THRESHPT is reached. After that level is achieved outputs are set to LOD for an amount of time defined by the hold timer. 10 Modulation Type: Threshold/Hold-on - IOSS="0": Modulation starts with channel outputs set to LOD until a lower threshold defined by THRESHPT is reached. After that level is achieved outputs are set to HOD for an amount of time defined by the hold timer. - IOSS="1": Modulation starts with channel outputs set to HOD for an amount of time defined by the hold timer. After that outputs are set to LOD until a low threshold is reached. 11 - IOSS="0": Modulation Type: threshold-low/Period. In this modulation mode channel outputs starts in LOD and a low threshold comparison is performed. When the threshold level defined by THRESPT is achieved, the channel outputs state changes to HOD until the modulation period expires. - IOSS="1": Modulation Type: threshold-high/Period. In this modulation mode channel outputs starts in HOD and a high threshold comparison is performed. When the threshold level defined by THRESPT is achieved, the channel outputs state changes to LOD until the modulation period expires.
5 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
6–7 SM	Sequencer Mode The SM[1:0] field defines the event that makes the channel address the next modulation word. This event can be a timer event or a threshold level event. Note that the channel does not necessarily increments the Modulation Word address thus the same Modulation Word is executed again in the new modulation sequence. NOTE: For time-out event selections, it is required the related time should be greater than 64 clock cycles. 00 Current modulation word is used, no advance is performed on any event. No timer is activated by this modulation word. 01 Advance to the next modulation word when a Timer time-out event is detected. The Timer is activated when the modulation word is accessed by the channel.

Table continues on the next page...

REACM2_MWBK n field descriptions (continued)

Field	Description
	10 Advance to the next modulation word when a Hold Timer time-out event is detected. The Hold timer is activated when the modulation word is accessed by the channel. This is only for MM=00 (THR-THR). 11 Advance to the next modulation word when a threshold level is achieved (greater or equal). The threshold level is defined by the THRESPT pointer.
8 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
9–11 HOD	High Output Drive The HOD[2:0] field defines the values driven on the chn_c, chn_b and chn_a channel output pins, respectively, when the channel is at ON state.
12 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
13–15 LOD	Low Output Drive The LOD[2:0] field defines the values driven on the chn_c, chn_b and chn_a channel output pins, respectively, when the channel is at OFF state.
16–17 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
18–22 THRESPT	Threshold Pointer The THRESPT Threshold Pointer is the address of the Threshold Bank that holds values to be used for ADC result comparisons in the modulation process. This pointer has the resolution for a 16-bit data stored in the register described in section “REACM Threshold Bank Register (REACM_THBK)”.
23 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
24–26 STPT	Shared Timer Pointer The STPT Shared Timer Pointer field is the pointer for the Shared Timer Bank. The Shared Timer is used as timer sequencer defining the advance of Modulation Words.
27–28 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
29–31 HDOFFTPT	Hold on/off Pointer The HDOFFTPT field is the address of the Hold timer value in the Hold Timer Bank that is used in the modulation cycle. Note that there are modulation sequences that does not require hold on/off measurement such as threshold/threshold modulations.

38.5 Functional Description

The following sections describe the Reaction Module functionality.

The Reaction Module is designed to allow a closed feedback loop control over the driver load currents. The load can be an injector for direct injection system or an electromagnetic actuator for robotized transmission. In both cases a solenoid is expected to be the actual load. The module architecture is based on shared resources sub-modules

that can be used by all 10 Reaction Channels. Please refer to [Figure 38-4](#) for details about the module architecture. The Reaction Module is composed of the following internal sub-modules:

- Reaction Channel
- Modulation Control Word Bank
- Shared Timer Bank
- Hold Time Bank
- Threshold Bank
- Period Pulse Generator

The following sections describe each one of these sub-modules and their combined operation.

38.5.1 Reaction Channel

The Reaction Channel is the core of the Reaction Module. Each channel controls three output pins and it is composed of a Control FSM which receives parameters to generate a modulated waveform as well as the timer control window usually provided by eTPU. The ADC interface sub-module indicates to the channel that a new ADC result is available. If *activated* by the eTPU channel, the Reaction Channel reads the Modulation Word which provides addresses for the Threshold Value bank, Shared Timer Bank and Hold Time Bank. The data provided by the Threshold Bank is compared with the incoming ADC result. Based on that comparison the Reaction Channel state machine selects the values for the output pin registers. The modulation only occurs if *activated* by the timer control signal, which can be generated by eTPU, or can be controlled by software by writing to the SWMC, software modulation control bit, in the Channel Configuration register. The MM and SM fields in the Modulation Word provide the Modulation Mode and Sequencer Mode control respectively. The following figure describes the internal architecture of the Reaction Channel and its interconnection with other sub-modules.

NOTE

The indexes corresponding to the register bit widths in the figure below will vary depending on the chip implementation. See the register descriptions for the correct bit width in your chip.

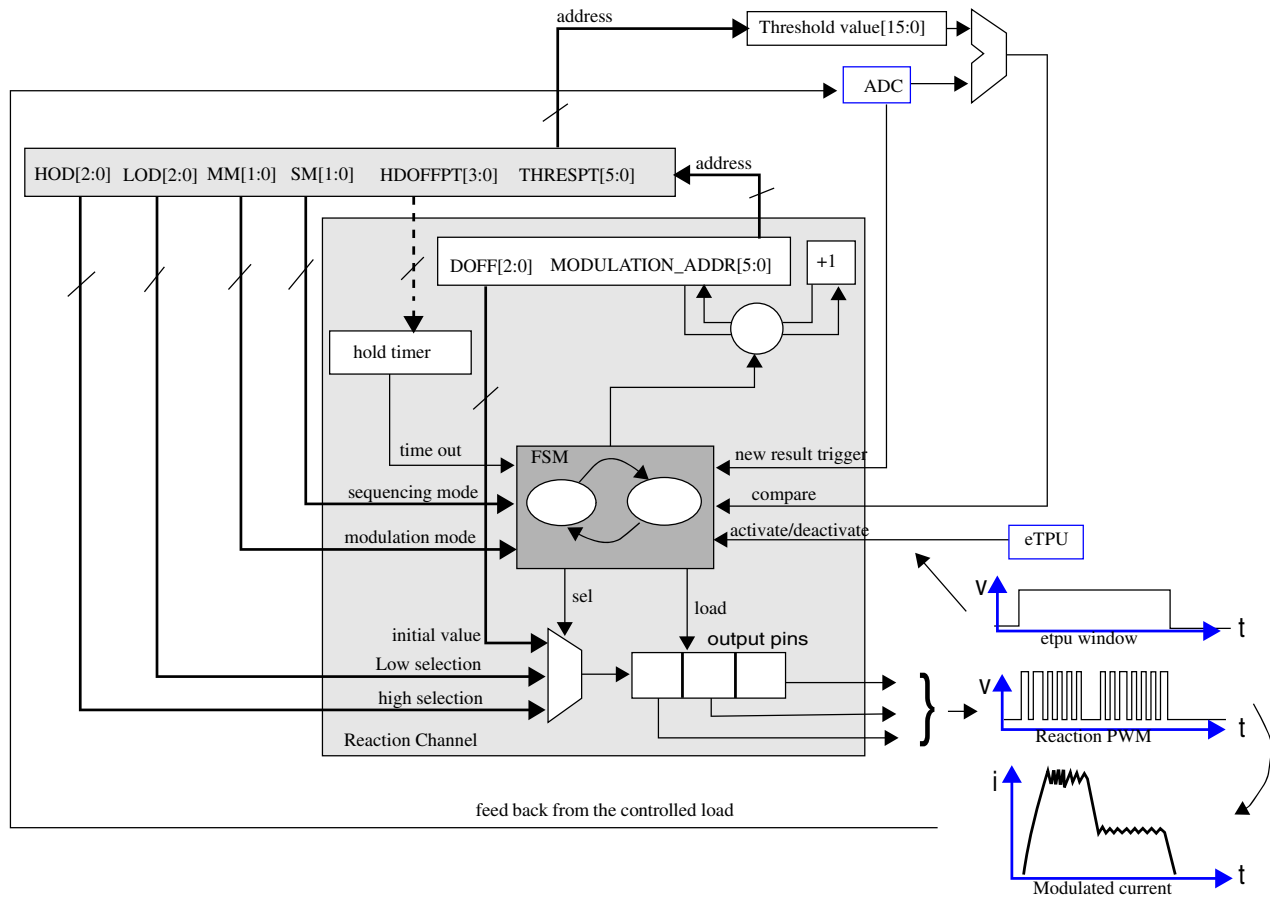


Figure 38-126. Reaction Channel Architecture Simplified Diagram

The addresses stored in the Modulation Word are pointers to timers and threshold banks. HOD and LOD provide the 3-bit field which controls the states, HIGH and LOW, respectively for the Reaction Channel output pins. The Reaction Channel FSM selects the values for the output pins as well as the load signal used to update those values. The modulation executed by the Reaction Channel is based on the feedback loop provided by the ADC results. These results are routed to one side of a comparator while the other comparator input receives the selected Threshold value from the threshold bank. The comparison result indicates for the Reaction Channel FSM if the output pin value should be defined either by HOD or by LOD. As a result a PWM is generated on the output pins. Note that since there are three independent outputs the PWM signal can be generated on any output or even on several outputs at the same time. Usually a PWM signal is generated in one of the outputs only, while the others outputs are used to control power supply switching, for example.

The preceding figure shows three waveforms at the right side of the figure. On the top there is the control signal from eTPU which defines the boundaries of the modulated waveform. In the middle there is the PWM waveform which is used to drive the power ON and OFF at the load, thus generating the third waveform which represents the current passing through the load.

NOTE

The modulation word sequence is limited to 16 words because the address value to the modulation control word bank can vary only from the configured initial value up to a maximum value given by (MODULATION_ADDR + 15).

38.5.2 Modulation Control Words Bank

The Modulation Control Word provides information about the modulation type to be performed and the addresses for several address banks. REACM2_MWBK_n describes the Modulation Word register and [Figure 38-127](#) describes the interfaces of this sub-module with some other sub-modules in the Reaction Module. The information stored in the Modulation Word are:

- Modulation control parameters for the Reaction Channel
- Threshold Value Register Bank address
- Hold time bank address
- Shared Timer bank address

All channels share the information stored in the Modulation Control Word Bank, which provides a size-effective implementation avoiding the duplication of information and allowing flexible implementation. The sharing of modulation control words allows several channels to execute the same modulation sequence.

The Modulation Control is designed to be used by all reaction channels as a centralized resource. However, only one channel is able to access the Modulation bank at a given time. Therefore, there is a priority in the selection of the channel that will have the access granted, but note that this condition does not occur too often since ADC results are provided for one channel at a time.

An arbiter in the Modulation Control Word bank selects one of the channels which are requesting access to a modulation word. The priority criteria is fixed and based on the channel number, considering Channel 0 the highest priority channel. The channel selected by the arbiter receives an acknowledge signal which indicates that channel was selected and therefore can access the modulation word.

Note

In order to avoid an initial delay when processing a timer window start event, the channel performs an speculative read operation of the first modulation word when it is enabled

(CHEN is configured). Therefore, when the modulation cycle is triggered by the timer window start event, all needed information for the modulation is already stored inside the channel.

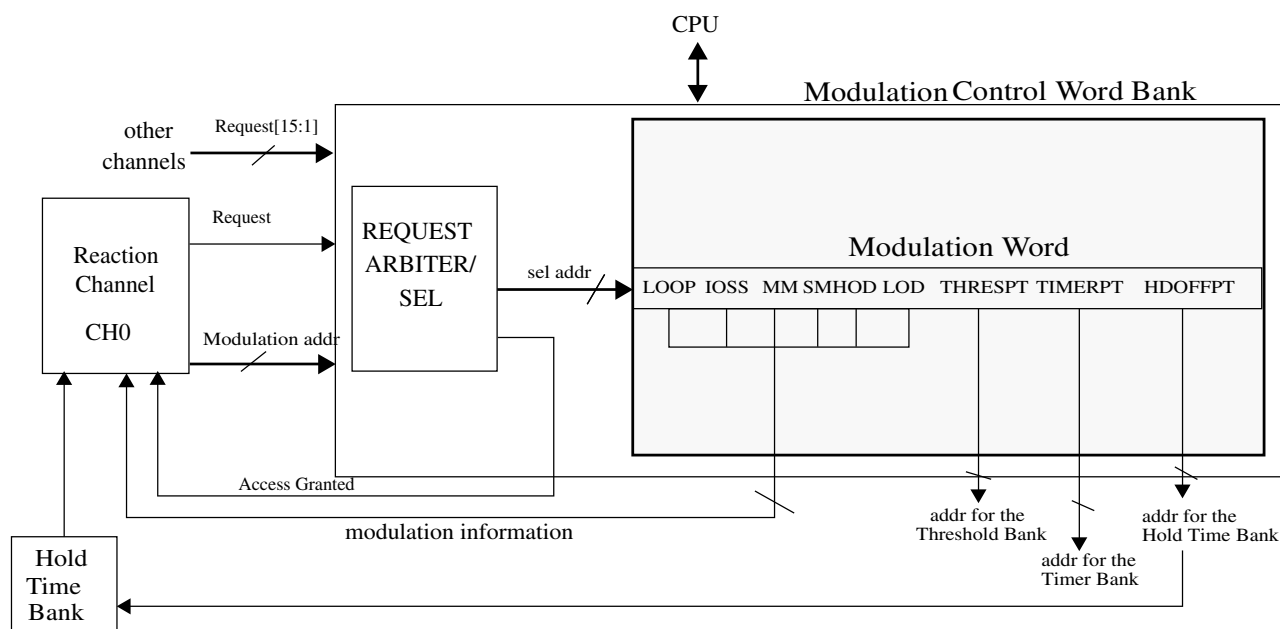


Figure 38-127. Modulation Control Word Bank Interfaces

38.5.3 Shared Timer Bank

The Shared Timer bank is an innovative concept of dynamic timer allocation. Since the number of timers can be smaller than the number of channels in the Reaction Module, there is a possibility that all timers are allocated at a certain time. This architecture is based on the low probability of such scenario since the timer allocation is a sporadic event. The timers in the Shared Timer Bank are usually in IDLE state until they are allocated by a channel. Any timer can be allocated as soon as it is in IDLE. The shared timers work in conjunction with a timer bank which stores values to be used by the timing measurement.

The Shared Timer Bank block is composed of two sub-modules:

1. Four 16-bit counters.
2. A bank with five selectable 16-bit time values.

If there is an attempt to allocate more than existing timers then an error flag TAER (see [Figure 38-134](#)) is set and no timer is allocated by the requesting channel. As a general guideline the system should be dimensioned in such a way that the timer allocation is always possible.

Note

In case of an allocation error the channel forces DOFF to its output pin preventing any damage to occur to the actuator being controlled. This state will not change until TAER bit is cleared.

During the timer allocation the channel also provides the TIMERPT pointer which selects a timing value. The value pointed by TIMERPT is loaded into one of the 4 counters which counts down until reaching zero. At this time a time-out indication is sent to the requesting channel and the timer is deallocated, moving back to IDLE state.

In case of several timer activation requests being issued at the same time, the logic in the Timer Bank will prioritize giving higher priority to the channel with lower number, thus channel zero has higher priority than the others. No flags are set in case several requests are issued simultaneously unless there are more requests than the number of available timers. In this case the TAER error flag is set in the requesting channel status register. The following figure presents a block diagram of the Timer Bank.

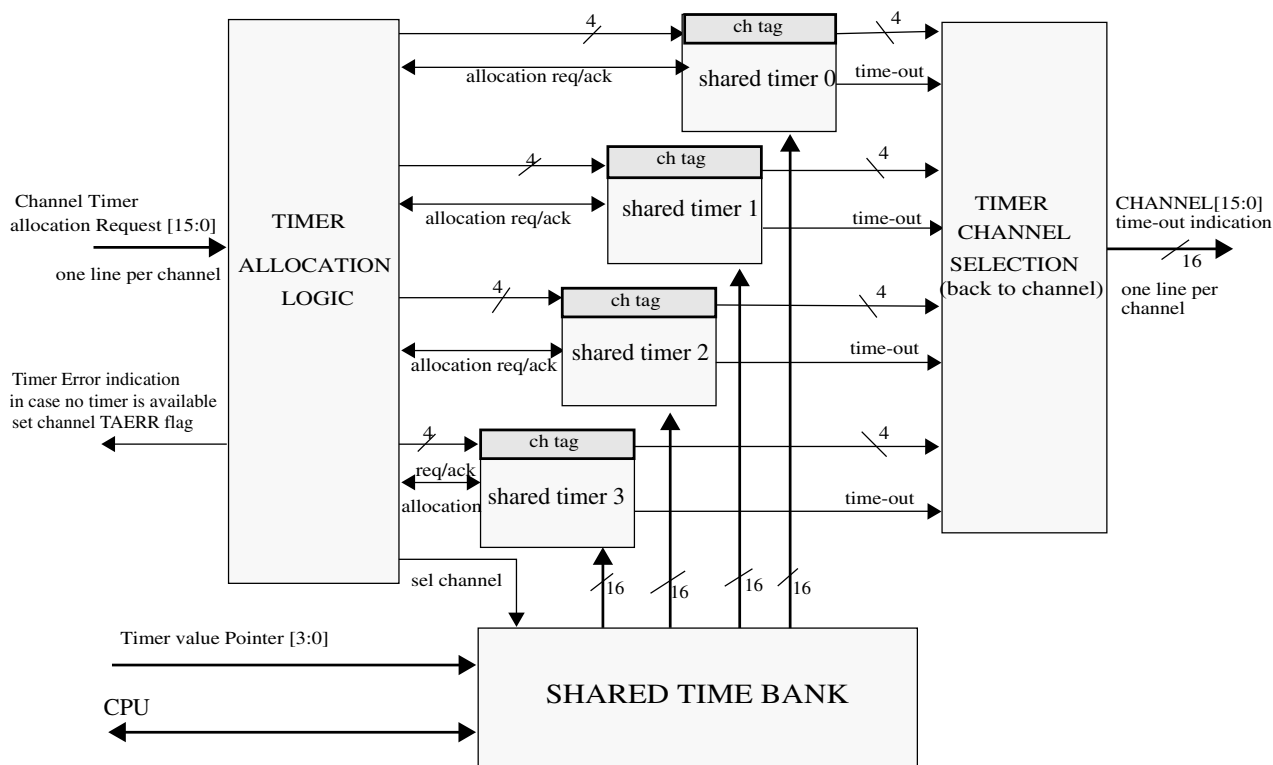


Figure 38-128. Shared Timer Bank Block Diagram with 4 Timers

38.5.4 Hold Time Bank

The Hold Time Bank stores time values that are used by the hold-on/off periods during the modulation process. This bank is shared among all channels and is addressed based on the HDOFFPT pointer in the Modulation Control Word. The following figure shows a block diagram of this bank and its interconnections.

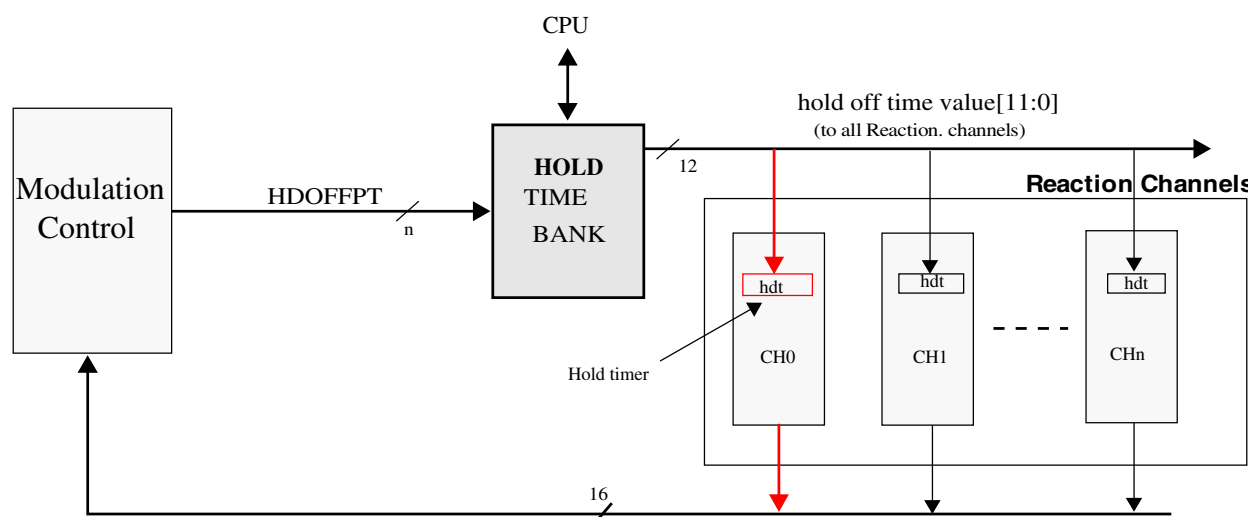


Figure 38-129. Hold Time Bank Block Diagram

The Modulation Control Word generates the address for the Hold Time bank which then generates the hold-on/off time value for the channels. The channel that addressed the Modulation Word captures the 12-bit value storing it in the channel Hold timer. In The preceding figure channel CH0 is requesting access to the Modulation Word thus receiving hold-on/off value from the Hold Time Bank. Each Reaction Channel has its own internal hold timer.

38.5.5 Threshold Bank and Comparator

The Threshold Bank and Comparator contains the threshold levels to be compared against the ADC results. This sub-module also performs the comparison between the ADC result and the selected threshold value.

The Threshold bank can be programmed with up-to 32 threshold values.

An ADC result received from the on-chip ADC is connected to the ADC router and then to the comparator (see the following figure). This result stays at the comparator input until the Reaction Channel and Modulation Control Word selects the address of the Threshold value. The THRESPT field in the Modulation Control Word sub-module is

used as the address for the Threshold Bank. Note that if Threshold-Threshold modulation mode is used two comparisons need to be executed with two different Threshold values. After the first comparison is made the THRESPT pointer is incremented and a new comparison is done:

- First comparison: $COMP = ADC\_DATA \geq THRESHOLD\_VALUE[THRESPT]$
- Second comparison: $COMP = ADC\_DATA \geq THRESHOLD\_VALUE[THRESPT + 1]$

The COMP, comparison result, is routed to the channel selected by the received TAG. Once receiving the comparison result the channel takes the appropriate actions in order to execute the modulation mode as defined by the Modulation Control Word.

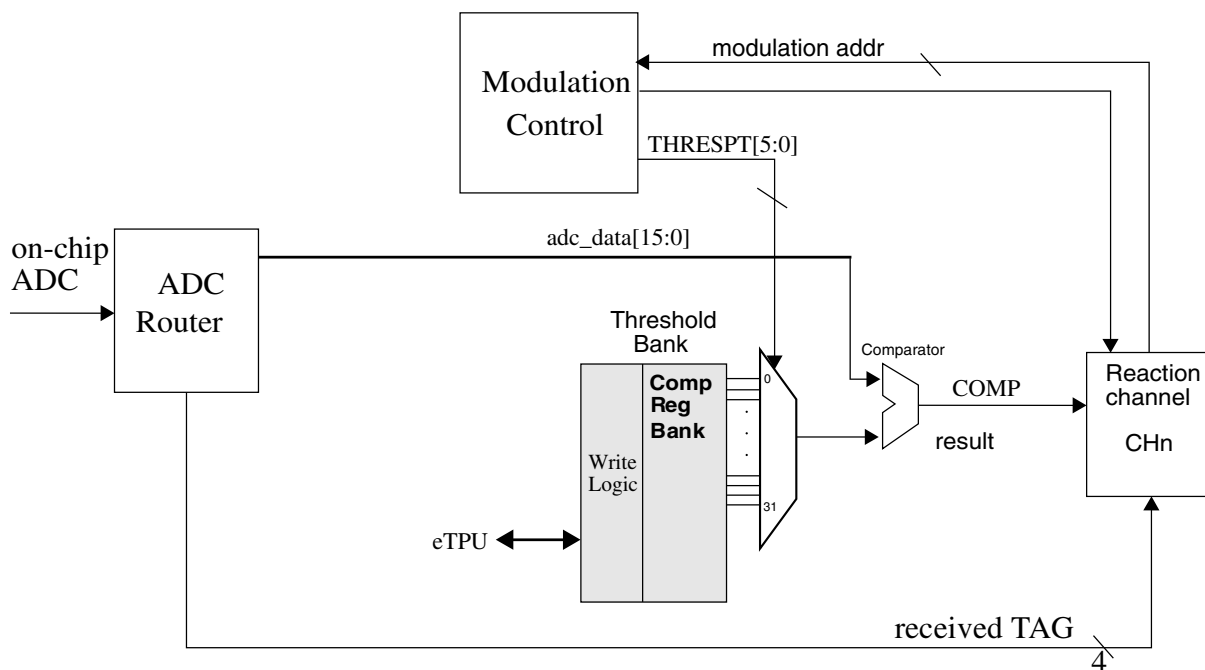


Figure 38-130. Threshold Bank and Comparator Block Diagram

38.5.6 ADC Interface

The ADC Interface connects to the eQADC module through a Parallel Side Port (PSI). Please refer to the eQADC chapter for more information about this interface. The eQADC sends conversion results to the Reaction Module which are received by the ADC Interface sub-module. The ADC Interface is capable of distinguishing between ADC results, Time stamps and *Prefill* information from eQADC. Only ADC results with no *Prefill* are considered by the Reaction Module as valid ADC samples. The TAG value received with the conversion result indicates which channel the result is addressed to. The

ADC interface is also responsible to indicate for the selected Reaction Channel that a new ADC result is available. It is also possible to access the ADC Interface input buffer through the REACM_SINR register.

38.5.6.1 Input Buffer Overrun

The OVR flag indicates than an input buffer overrun occurred. This buffer is normally written by the ADC interface (PSI) but can be written also by the CPU using the REACM_SINR register.

The ADC Interface data (ADC_TAG and ADC_RESULT) should not be updated until all Reaction Channels (with ADCR = ADC_TAG) process the received data. The OVR flag is set in the case of an overrun condition occurs, indicating that at least one ADC data was lost (the input data that caused the overrun is lost).

The following situations can generate an input overrun:

- Two consecutive sample data are received from ADC or from CPU.
- ADC and CPU sending sample data at the same time (asynchronous events).
- Input buffer is holding a sample data that is being processed by some channel and a new sample is received from ADC or CPU (normal mode of operation).

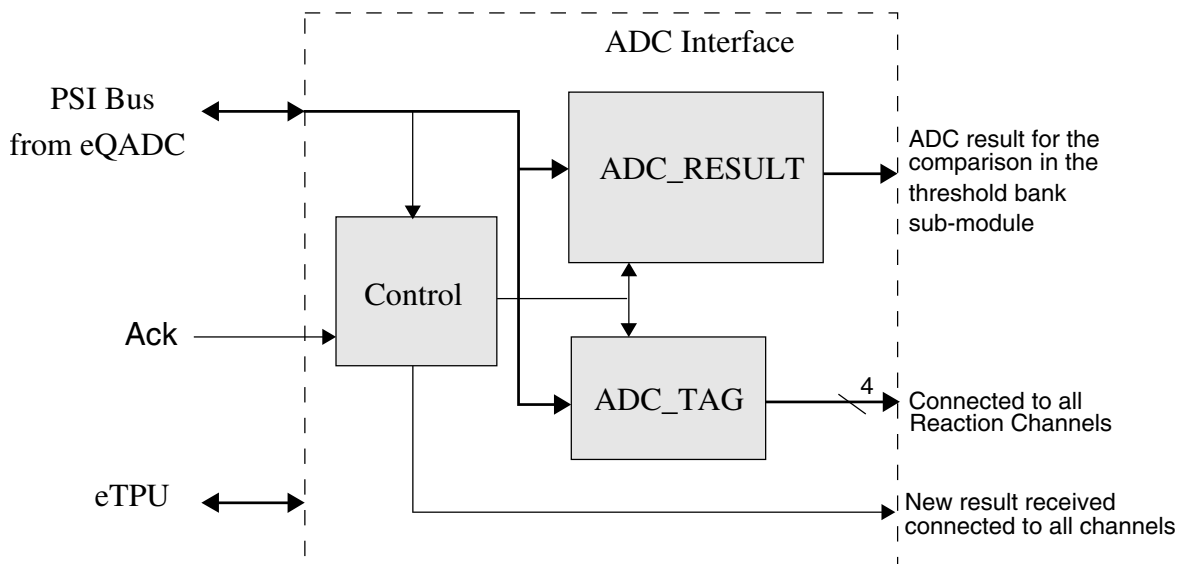


Figure 38-131. ADC Interface Block Diagram

The maximum throughput supported by the Reaction Module depends upon the TAG of the incoming ADC data. If only one Reaction Channel is addressed by the ADC TAG then the maximum supported ADC data rate is 1 sample for each 5 clock cycles. If two Reaction Channels are addressed by the same ADC TAG, thus having the same CHRR

ADCR field and are active at the same time, the maximum supported ADC data rate is 1 sample for each 10 clock cycles. In general if (n) channels share the same TAG and are active at the same time the maximum supported ADC data rate is 5 x (n) clock cycles.

These limitations are related to the sharing of internal Reaction Module resources such as the Modulation Word bank. The ADC conversion data should remain stable in the ADC interface until it is used by all channels with matching TAG and CHRR ADCR fields.

Note that if all active channels have different CHRR ADCR fields, i.e. are assigned to different TAG, the maximum supported ADC rate is 5 clock cycles. If multiple channels have the same CHRR ADCR field but only one is active at a time, then the maximum supported ADC data rate is also 5 clock cycles.

38.5.6.2 On-the-fly ADC data Acquisition

The ADC Interface and the Threshold bank can operate in a learn mode, meaning that the received ADC result can be stored in the Threshold Bank and used for comparisons. This functionality allows the user to interactively calibrate the modulation levels or capture on-the-fly levels from the external application board for debug purposes. The THRR register is used for that purpose. See [Figure 38-128](#) for more details about the register architecture. [Figure 38-132](#) describes the main connections between the ADC interface and Threshold Bank sub-modules.

The routing of received ADC results to be stored in the Threshold Bank is independent from the routing of the same result to the Reaction Channels (see [Figure 38-135](#)). Thus the same ADC result can be used in a comparison and stored in the Threshold Bank.

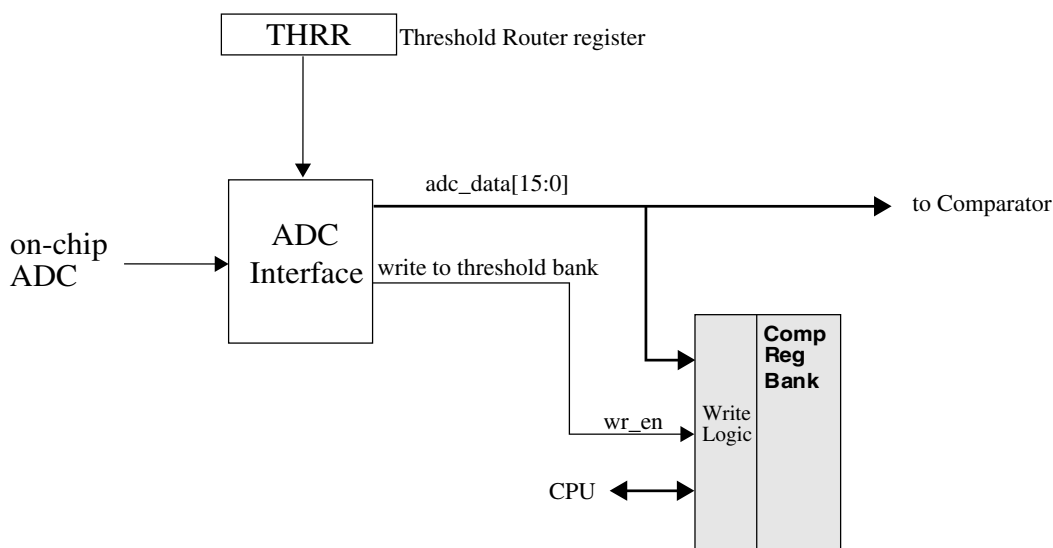


Figure 38-132. ADC interface and Threshold Bank interconnections

38.5.7 Prescalers

The Prescalers provide internal system clock divided signals to be used by internal timers. There are 2 prescalers in the Reaction Module. A 12-bit prescaler HPRE[11:0] and an 8-bit prescaler TPRES[7:0]. Both are defined in the Timer Configuration register TCR. The HPRE[11:0] is dedicated to the Hold timers within the Reaction Channels. The TPRES[7:0] is used by the Shared Timer Bank counters. The HPRE[11:0] and TPRES[7:0] prescalers are enabled by HPREN and TPREN bits, respectively, in the Module Configuration register. Note that prescalers operate in a similar way regarding its activation. Once the prescaler is enabled by HPREN or TPREN bits in the Module Configuration Register it starts a new count sequence meaning that it is put in reset state and will generate the first prescaler tick after it reaches the programmed value defined by HPRE or TPRES fields.

38.5.8 Period / Delay Generator

This signal generator provides periodic pulses to support multi-channels operating in the threshold-period modulation modes (see [Threshold low/Period Mode](#) and [Threshold high/Period Mode](#)). There is an exclusive prescaler counter for period / delay counters. The modulation period is defined by the PER[15:0] bits in the Period Configuration register and supplied by internal period counter. This counter can be disabled if the field PER[15:0] is all 0.

Each channel receives a period pulse signal delayed from the others. The delay of the period signals between consecutive channels is defined by the field DLY[15:0] in the Period Shift Delay Configuration register. Channel 0 receives the period signal without delay. Channel 1 gets the channel 0 period signal delayed by DLY[15:0], channel 2 gets channel 1 signal delayed by the same amount, and so on. However, if the delay is zero, all period pulses occur at the same time.

Note

The correct period and delay times depend on the period prescaler setting.

The period pulses are generated for all available channels regardless if the threshold/period modulation mode is being executed. For instance, if channel 0 and 2 have Threshold/Period and channel 1 doesn't, the delay from 0 to 2 is "2 x DLY".

The following figure shows the architecture of the Period / Delay generator sub-block.

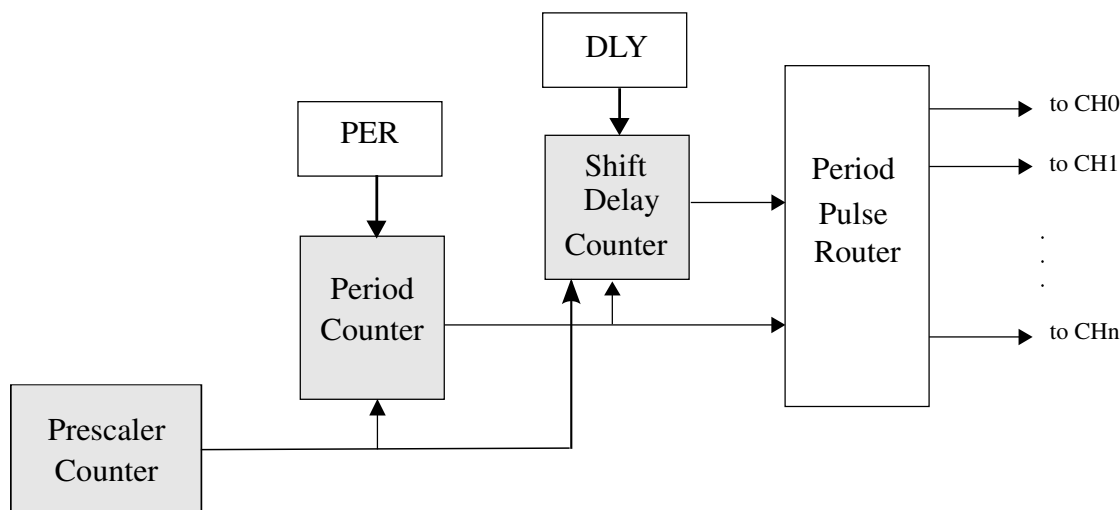


Figure 38-133. Period / Delay generator sub-block

38.5.9 Banked Mode Support

Banked Mode is a Reaction Module hardware configuration which allows the sharing of Reaction Channel output pins at the device I/O level. The banked mode architecture allows the stacking of up to four Reaction channels. The following figure shows the connection between two adjacent channels, CH0 and CH1. The CCR BSB bits are used to control the configuration of channel output logic. Thus if BSB[0] in Reaction Channel [0] is asserted to 1 and the channel is not in the active state, ch0_a output will be switched from CH[0] OUT[0] to CH[1] OUT[0]. If a channel is active, i.e. executing a modulation, it takes control over its outputs independent of BSB bits setting. The banked mode logic is extended to Reaction Channels CH[2] and CH[3] thus defining a group of 4 channels. For simplification the following figure shows only channels CH[0] and CH[1] logic.

Note

When CHEN=00 for CH0 (channel disabled), the BSB bits don't influence the channel output, which is driven by CH0's DOFF. Therefore, a banked injector driven by CH1 will have part of its controls off, even if CH1 is enabled.

To use BSB of CH0 in this case, an option is to program channel CH0 with CHEN=11 (channel enabled) and with SWMC=0 (modulation OFF).

Note

In case of using CH2 in banked mode with CH0, the intermediate channel CH1 should also be configured with CHEN different from 00.

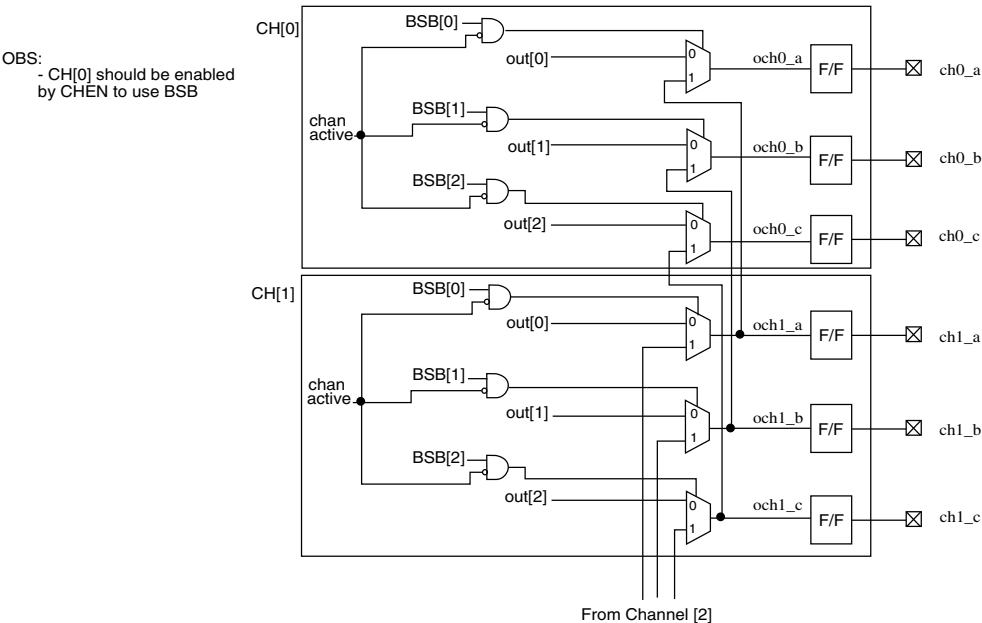


Figure 38-134. Banked Mode showing stacking of channels [0] and [1]

The Banked Mode support hardware is implemented on groups of up to 4 channels. The groups are defined as CH[3:0], CH[7:4], and CH[9:8].

Table 38-125. Banked Mode groups

CH3 → CH2 → CH1 → CH0
CH7 → CH6 → CH5 → CH4
CH9 → CH8

Thus CH[3], CH[7], and CH[9] do not connect to the subsequent channel which are CH[4], CH[8], and CH[0] respectively.

38.6 Modulation Modes

This section describes the modulation modes provided by the Reaction Module.

38.6.1 Threshold/Threshold Mode

This mode is programmed by using $\text{REACM2\_MWBKx[MM]} = 00$. In this modulation mode the Reaction Channel tries to maintain the ADC results between two Threshold limits. The modulation actions are as follows:

- $\text{ADC result} \geq [\text{THRESPT}]$: The Reaction Channel turns the outputs off by loading $\text{LOD}[2:0]$ to the channel outputs.
- $\text{ADC result} < [\text{THRESPT} + 1]$: The Reaction Channel turns the outputs on by loading $\text{HOD}[2:0]$ to the channel outputs.
- $[\text{THRESPT}+1] \leq \text{ADC result} < [\text{THRESPT}]$: The Reaction Channel keeps the outputs unchanged.

The following figure indicates the threshold/threshold modulation mode.

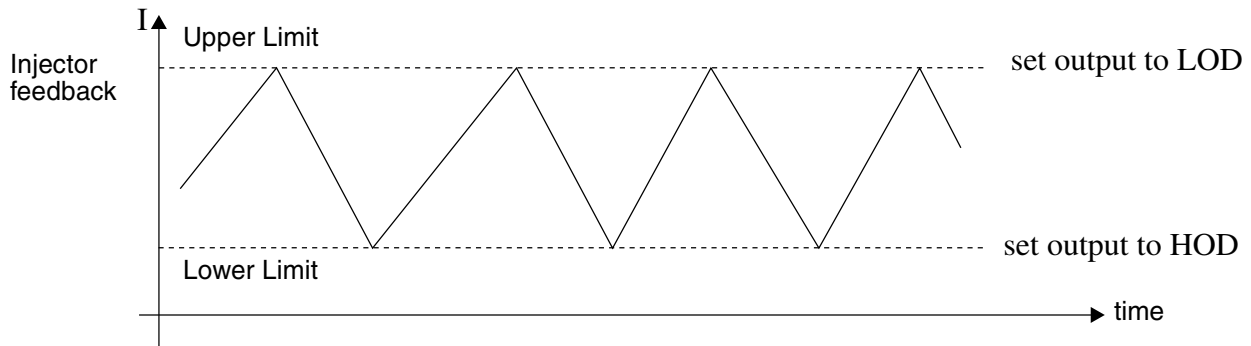


Figure 38-135. Threshold/Threshold Modulation Mode

There can be overshoots or undershoots in the real application related to the threshold limits. This occurs due to following:

- Feedback values are periodically sampled thus can present gaps on the measured values. For example when the sampling has been executed the value may have already passed the range defined by the limits;
- There is a delay between the sampling of feedback value and the reaction of the Reaction Channel. This is mainly caused by the ADC conversion time and its maximum sampling frequency.

The short circuit and open circuit detection are performed by monitoring the width of the ON pulse. For more details refer to [Monitored Modulation](#).

38.6.2 Threshold/Hold-off Mode

This modulation mode is programmed by using $\text{REACM2\_MWBKx[MM]} = 01$. In this mode the output pins are driven with $\text{HOD}[2:0]$ until the ADC results matches or is greater than the programmed Threshold. The output controls are then driven with $\text{LOD}[2:0]$ for a fixed amount of time defined by the hold timer. The initial modulation behavior can be controlled by using the $\text{REACM2\_MWBKx[IOSS]}$ bit as follows:

- When $\text{REACM2\_MWBKx[IOSS]} = 0$, modulation starts with outputs driven with LOD until the hold timer expires.
- When $\text{REACM2\_MWBKx[IOSS]} = 1$, modulation starts with outputs driven with HOD until the upper threshold level is reached.

Note that the upper threshold limit and the hold timer pointers are indicated by the Modulation Word in the REACM2_MWBK register.

Note

It is considered that in hold-off time the ADC samples are not reliable, therefore no comparison for MAXL or for output definition are performed. If these comparisons are necessary, the Threshold / Threshold modulation mode can be used.

The short circuit and open circuit detection are performed by monitoring the width of the ON pulse. For more details refer to [Monitored Modulation](#)

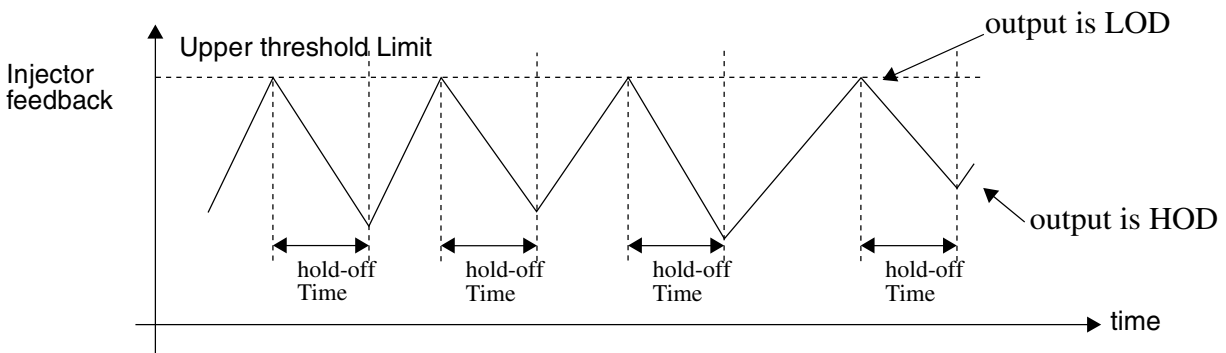


Figure 38-136. Threshold/hold-off Modulation Mode

38.6.3 Threshold/Hold-on Mode

This modulation mode is programmed by using $\text{REACM2\_MWBKx[MM]} = 10$. In this mode the output pins are driven with $\text{LOD}[2:0]$ until the ADC results is smaller than the programmed Threshold. The output controls are then driven with $\text{HOD}[2:0]$ for a fixed

amount of time defined by the hold time. The exception for this defined ON time occurs if the value of an input sample surpasses the MAXL. In this case the output pins change to LOD[2:0] immediately.

The initial modulation behavior can be controlled by using the REACM2_MWBKx[IOSS] bit as follows:

- When REACM2_MWBKx[IOSS] = 0, modulation starts with outputs driven with LOD until the lower threshold level is reached.
- When REACM2_MWBKx[IOSS] = 1, modulation starts with outputs driven with HOD until the hold timer expires.

Note that the lower threshold limit and the hold timer pointers are indicated by the Modulation Word in the REACM2_MWBK registers.

The open circuit detection is performed by monitoring the minimum width of the OFF pulse. The short circuit is indicated by monitoring the ADC samples if the input sample surpasses the MAXL value. For more details refer to [Monitored Modulation](#).

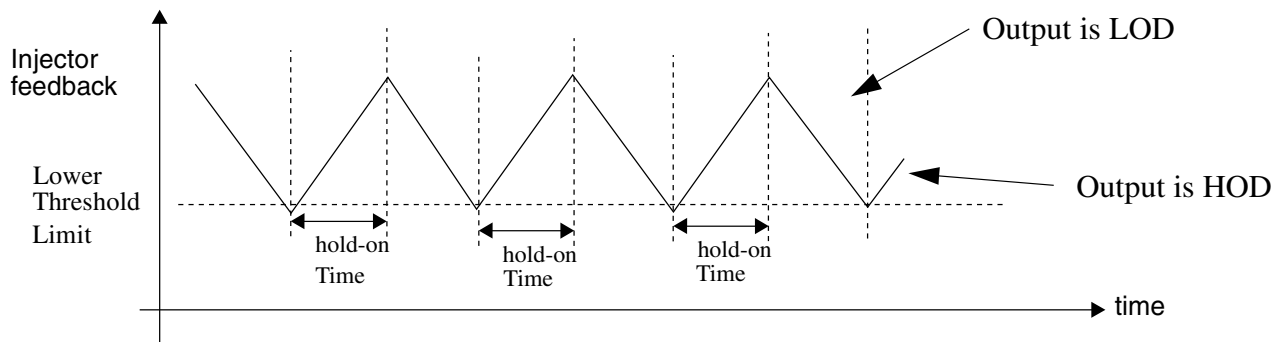


Figure 38-137. Threshold/Hold-on modulation mode

38.6.4 Threshold low/Period Mode

This modulation mode is programmed by using REACM2_MWBKx[MM] = 11 and REACM2_MWBKx[IOSS] = 0. In this modulation mode channel outputs start with LOD[2:0] and a low threshold comparison is performed until the ADC result is smaller than the programmed threshold. When the threshold level defined by THRESPT is achieved, the channel outputs states change to HOD[2:0] until the modulation period expires. The exception for this defined event occurs if the value of an input sample surpasses the MAXL. In this case the output pins change to LOD[2:0] immediately. Note that the lower threshold limit is indicated by the Modulation Word in the REACM2_MWBK registers, but the period time is given by the Period / Delay generator sub-block.

The following figure indicates the threshold low/period modulation mode.

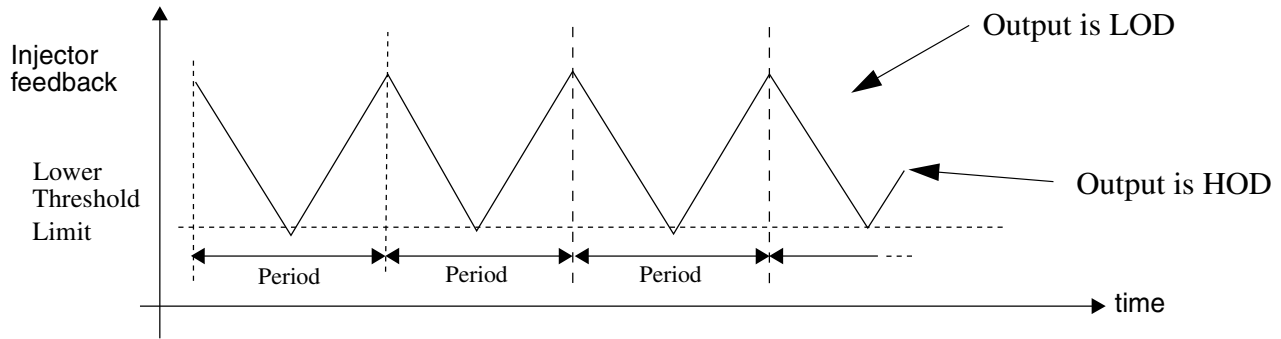


Figure 38-138. Threshold low/Period modulation mode

The open circuit detection is performed by monitoring the minimum width of the OFF pulse. The short circuit is indicated by monitoring the ADC samples if the input sample surpasses the MAXL value. For more details refer to [Monitored Modulation](#).

38.6.5 Threshold high/Period Mode

This modulation mode is programmed by using $\text{REACM2\_MWBKx[MM]} = 11$ and $\text{REACM2\_MWBKx[IOSS]} = 1$. In this modulation mode channel outputs starts with HOD[2:0] and a high threshold comparison is performed until the ADC results matches or is greater than or equal to the programmed threshold. When the threshold level defined by THRESPT is achieved, the channel outputs state change to LOD[2:0] until the modulation period expires. Note that the high threshold limit is indicated by the Modulation Word in the REACM2_MWBK registers, but the period time is given by the Period / Delay generator sub-block.

The following figure indicates the threshold high/period modulation mode.

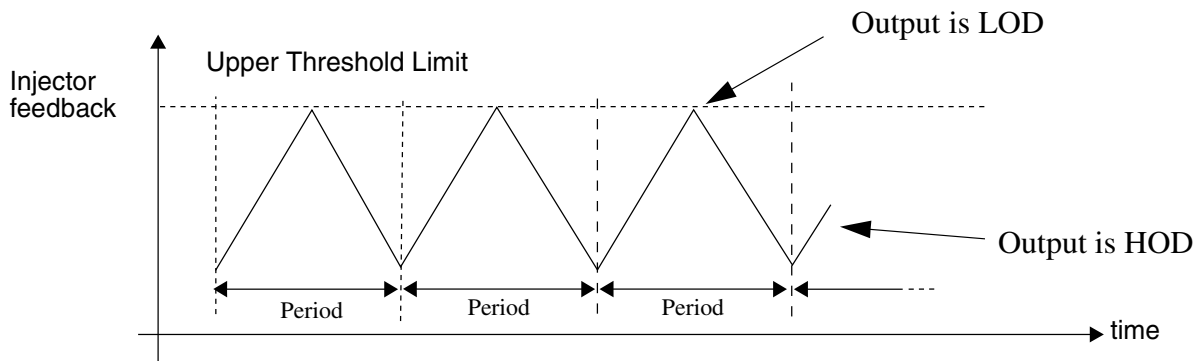


Figure 38-139. Threshold high/Period modulation mode

The short circuit and open circuit detection are performed by monitoring the width of the ON pulse. For more details refer to [Monitored Modulation](#).

38.6.6 Limitations on the Modulation Process

This section describes the Channel limitations on the modulation process such as the width and distance between consecutive modulation pulses.

38.6.6.1 Minimum distance between consecutive Timer Control Pulses

The control signal generated by the external timer which controls the modulation process in a worst case scenario has a minimum distance of 40 system clock periods. This is required for the Reaction Channel state machine to properly re-start the modulation process and address the correct modulation word. The worst case scenario considers a Reaction Module with 10 channels and all channels being activated at the same time. If this minimum distance is violated the modulation on the second pulse will not be executed, meaning that the channel output will be defined by the DOFF field.

Note

No error flag is set if this violation occurs.

The following figure describes the channel behavior if the minimum time between two consecutive timer control pulse is violated.

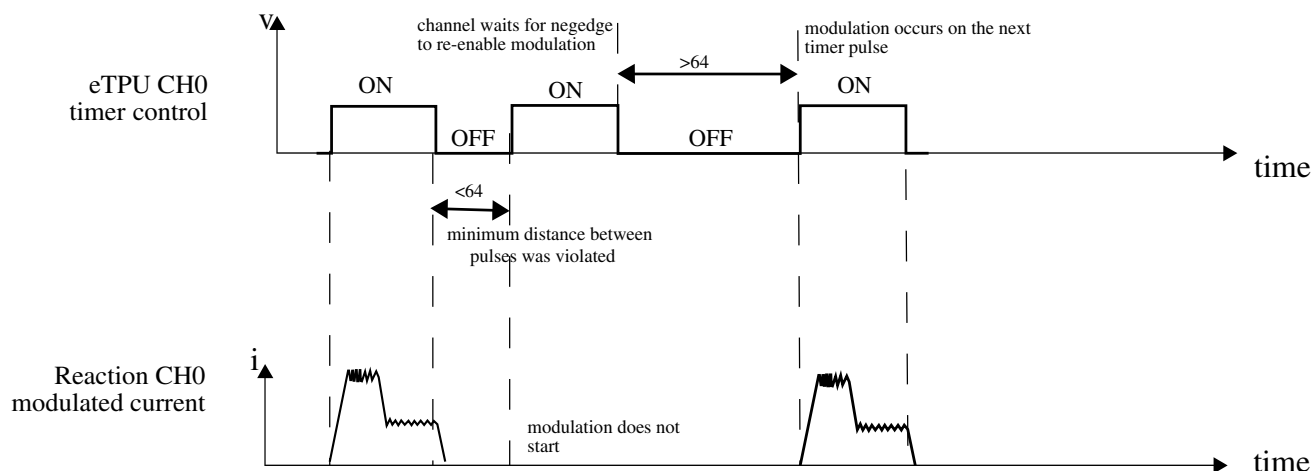


Figure 38-140. Limitation on the OFF modulation timing

38.6.6.2 Minimum Timer Control Pulse Width

There is no limitation on the Timer Control pulse minimum width. The channel just finishes the modulation process when the pulse ends even if the Modulation Words did not complete the expected sequence. Thus if a modulation phase is designed to have four Modulation Words starting from word 0, and the Timer Pulse ends at the second word (word 1) then the modulation process finishes at this point in time. The channel output is set to DOFF one clock cycle after the Timer Pulse ends. And the flag SQER is set because the SM bit field for the second modulation word is not 00. This flag occurs to indicate that the modulation was ended before the last phase of the sequence that uses the modulation word 3 and has SM=00.

The following figure describes the channel behavior when an early end of the Timer Control pulse occurs. Note that the channel output is driven to DOFF which causes the modulation to end. Note that an early end of pulse only affects the current modulation cycle. Meaning that on the next modulation cycle the modulation word 0 is executed first and all subsequent words are executed in the appropriate sequence.

Note

In some applications the modulation runs continuously and the input timer control signal is not used as a modulation pulse control but only as an enabling signal. Therefore, if the modulation is turned off, the REACM2 can issue a Modulation Word Sequence Error by setting SQER flag. In this case, this SQER flag can be ignored (masked) without prejudice.

An option, if the application permits, is to disable the REACM2 channel (CHEN=00), thus avoiding unwanted SQER error.

Modulation Modes

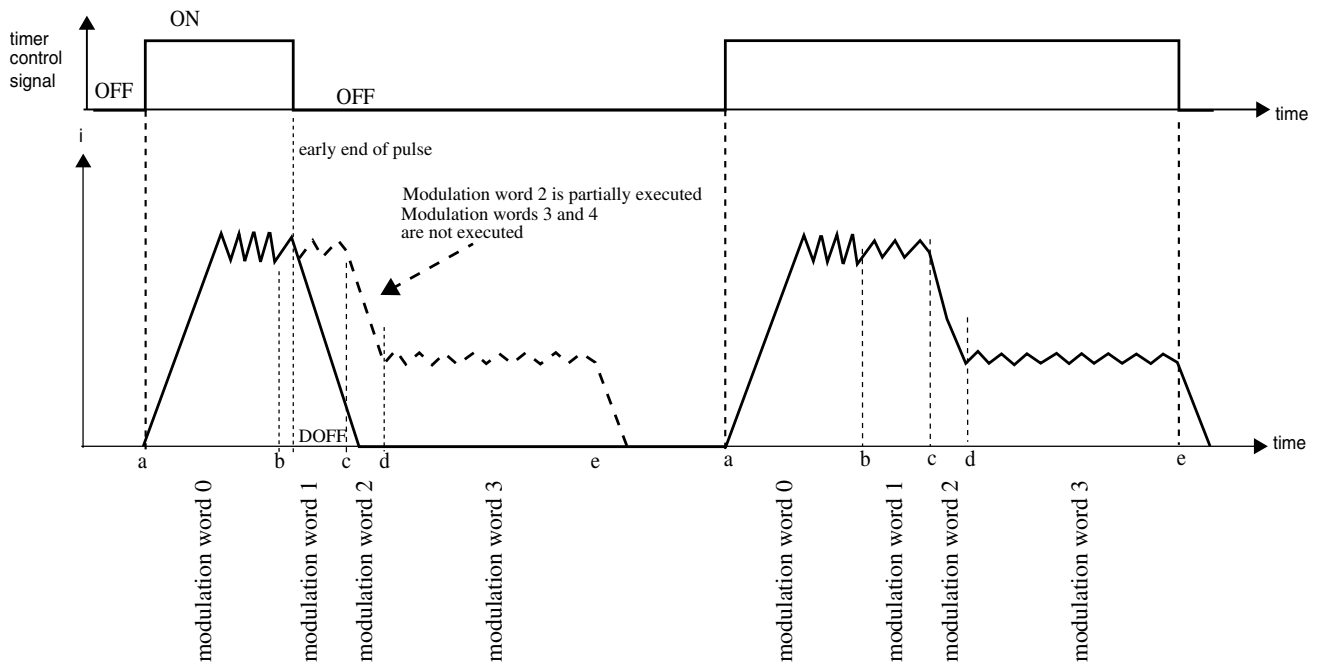


Figure 38-141. Early end of Timer Control pulse

38.6.6.3 CHOFF behavior during modulation

The CHOFF, channel output disable bit in REACM2_CHCRn, is intended to disable the channel immediately before a modulation cycle had ended. This is important on error conditions detected by the software, thus setting the channel outputs to a safe state defined by DOFF. It is important to note that re-activating the channel by setting CHOFF=0 does not imply that the channel outputs immediately return to the state defined by HOD and LOD on the modulation word being executed. Instead, the HOD and LOD are driven to the channel outputs only when one of the following conditions occur:

- A sequence advance event (time-out or threshold, depending on SM)
- A new sample is received (no matter whether the comparison matches or not)
- A hold-on/off time-out
- A period pulse occurs for the related modulation mode

38.6.6.4 Module initialization

In order to properly execute the modulation process the Reaction Module must be initialized with a correct sequence. One alternative is described as follows:

1. Make sure eTPU channels are at zero, inactive state
2. Setup eQADC module
3. Program the Modulation Words
4. Program the Threshold Bank
5. Program the Shared Timer Bank
6. Program the Hold Time Bank
7. Program Timer Configuration register
8. Program the Module Configuration register
9. Program the Period Shift Delay Configuration register
10. Program the Period Generator Configuration register
11. Program the Channel Router register
12. Program the Channel Configuration register
13. Start eTPU channels

It is important to note that the channel activation, by setting CHEN=01, should be done after all other registers were configured and before the input timer control signals are active. Violating this order may lead to errors when the modulation cycle is executed by the channel.

Note

If a glitch is introduced in the input timer control pulse, the channel stops to modulate and does not operate during the pulse just after the glitch. The glitch value for wrong operation ranges from 1 system clock to about 5 times the number of channels.

38.6.6.5 Potential glitches on the reaction channel outputs

Reaction channels outputs are digital signals that are controlled by several events that are not synchronized to each other. Due to this nature, there are a few cases when a glitch can occur at the channel outputs. If the channels outputs are going to be used by systems that are very slow compared to the generated PWM, these cases of glitches are not a problem.

The known cases of potential glitches are listed below:

- At the end of a modulation phase when a new modulation word will be used, a glitch can occur if the output has just changed prior to the end of the phase and the IOSS of the next phase forces a value that is the opposite of the last value.
- At the end of the whole modulation when the modulation is going to be disabled by the timer channel, a glitch can occur if the output has just changed to HOD prior to the disabling.
- If the threshold / threshold modulation mode is used with SM set to threshold (high) mode, at the end of the phase the output is switched off to LOD. If the modulation word for the next modulation phase has IOSS set to 1, a LOW glitch is generated. This can be avoided by setting HOD=LOD in the described previous modulation phase.

38.7 Monitored Modulation

The modulation executed by the Reaction Channel can be monitored by measuring the width of the PWM ON pulse or the OFF state.

38.7.1 Monitoring the ON Pulse Width

The modulation executed by the Reaction Channel can be monitored by measuring the width of the PWM pulses provided by the channel. If the pulse becomes too narrow it means that the load impedance is probably too low, thus indicating a possible short circuit. If the PWM pulses become too wide it may indicate an open circuit on the solenoid. These criteria can be used for threshold-threshold, threshold-hold-off, and threshold high-period modulation modes.

The PWM pulse is considered to start (rising edge) in the following cases:

- in the start of the modulation phase with IOSS=1 (threshold-threshold, threshold-hold-off, or threshold high-period modulation modes);
- when the hold-off time is over in the threshold-hold-off modulation mode;
- when the received sample is lower than the threshold LOW value (threshold-threshold, or threshold-hold-off modulation modes);

The PWM pulse to be monitored ends (falling edge) when the input sample value surpasses the threshold HIGH in the above modulation modes.

NOTE

When modulation word progresses to new modulation phase the current pulse width monitoring is stopped and a new one is started if necessary. Therefore there is no pulse width extension nor narrow pulse occurrence due to modulation phase progress.

The short detection has an additional configuration to set the SCDF flag. The SCDF flag is set if a single detection event occurs (REACM2_CHCR register field SDCM=0) or if two short detection events occur in a row (SDCM=1). For simplicity, the following descriptions are for single short detection case.

The limits for narrow and wide pulses are defined by RANGE_PWD and MIN_PWD registers (see [Figure 38-140](#) and [Figure 38-141](#)). These values apply to all Reaction Channels. The PWM pulses are measured by the Hold timers within the Reaction Channels. This is possible only during idle periods of this timer, for example from the moment a hold-off time-out occurs until the maximum threshold is reached when HOD is being used for the channel output pins. During this period the hold counter is not used and thus it can measure pulse widths and compare them against pre-defined limits defined by RANGE_PWD and MIN_PWD registers, as shown in [Figure 38-142](#).

Note

Consider an uncertainty of (+1) in the value MIN_PWD and (MIN_PWD + RANGE_PWD) when calculating the pulse width limits. The Hold timer prescaler contributes to this uncertainty.

For a programmed MIN_PWD value, a pulse wider than (MIN_PWD+1) does not set the SCDF flag.

For a programmed (RANGE_PWD + MIN_PWD) value, a pulse narrower than or equal to (RANGE_PWD + MIN_PWD + 1) does not set the OCDF flag.

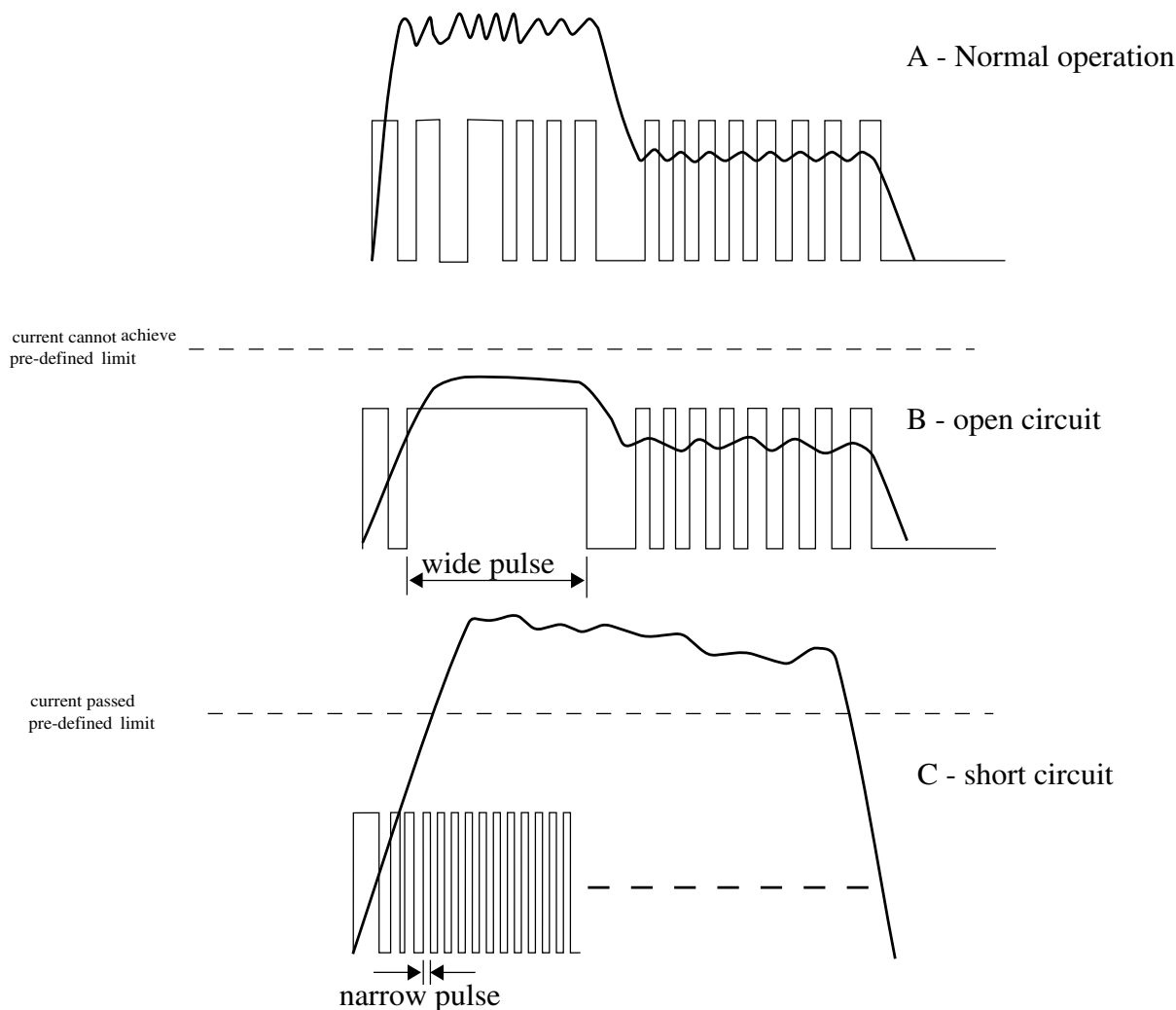


Figure 38-142. Fails detected by the pulse width monitoring

Figure 38-143 describes in more detail an open circuit detection using the Hold timer. Note that the Hold timer is used when the output is at the ON state, which means HOD is used to drive the outputs, thus hold-off functionally is not required at this moment. The Hold timer is loaded with MIN_PWD and starts counting. After a time-out occurs RANGE_PWD value is loaded and the Hold counter starts counting again. An open circuit is detected if the timer times-out on the second RANGE_PWD counting and the PWM pulse is still high, meaning that the outputs did not switch to LOD. In this case the OCDF flag is set in the Channel Status Register (REACM2_CHSR), of the corresponding channel.

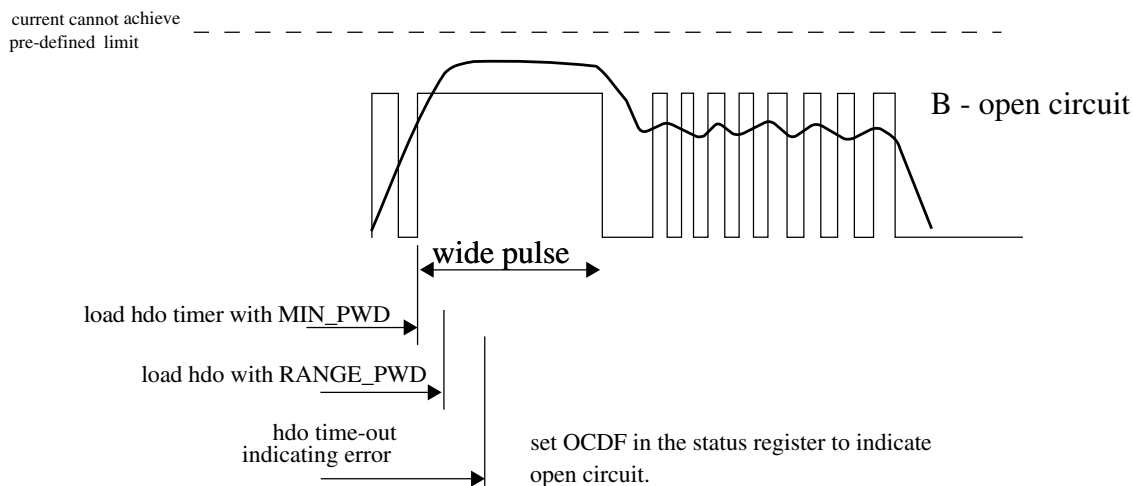


Figure 38-143. Open Circuit detection using Hold timer

Figure 38-144 describes how the Hold timer detects a short circuit. The Hold timer is loaded with the MIN_PWD value and enabled. The condition indicating that a short circuit occurred is that the Hold timer is still running and LOD is loaded to the channel output pins. That means the PWM pulse is too narrow and a short circuit has occurred. In this case the SCDF flag is set in the Channel Status Register (REACM2_CHSR) of the corresponding channel.

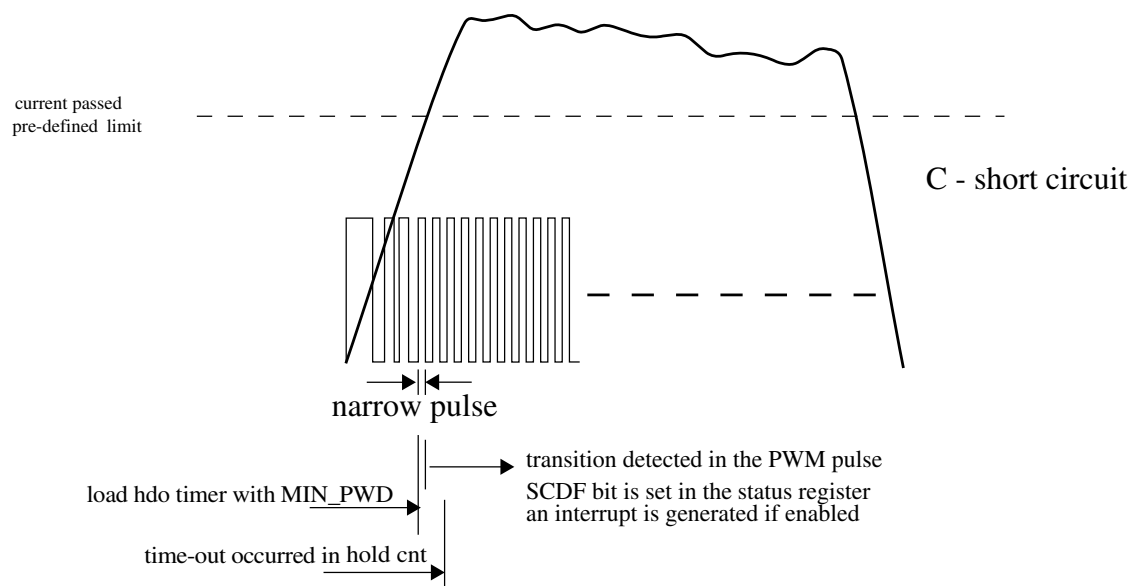


Figure 38-144. Short Circuit detection using Hold timer

Note

In order to define RANGE_PWD value it is required to consider that the Hold timer already measured MIN_PWD, thus actually the maximum allowed pulse width = (MIN_PWD + RANGE_PWD). In other words, RANGE_PWD = (maximum allowed pulse width - MIN_PWD).

IF MIN_PWD = 0x00 or RANGE_PWD = 0x00 no pulse width is performed.

The CHSR SCDF flag does not set if the pulse was finished by disabling the modulation (i.e. eTPU channel=0 or SWMC=0) or by disabling the channel, CHEN=00, even if it ended shorter than MIN_PWD. However this flag can set in some situations that really indicates a short pulse detection but it is the result of some internal condition of the reaction module. The known situations are listed below:

- When a modulation phase with MM=01 (threshold / holdoff) starts with IOSS=1 with the previous modulation phase set to SM=11 (advance on threshold).

The CHSR OCDF flag only sets when the channel is enabled (CHEN not null) and the eTPU channel signal or SWMC is active too. However, the OCDF flag can set in some cases when the CHOFF bit is set. In this case, this OCDF flag should be disregarded because it is a false indication of the detector.

There can be a conflict of resource allocation if the Hold timer is used as the timer for the sequencer mode (SM=10) because it is not possible to have the detection of minimum and maximum pulse widths. This configuration is only allowed to be used with the threshold-threshold modulation mode but the monitor of short / open circuit is not enabled. The RAER flag will be set for any other modulation mode.

NOTE

An application case is to use SM=10 with threshold / threshold modulation mode and IOSS=0. The channel output starts OFF and the SM hold timer can be configured to have a time-out before the threshold low is reached. If the threshold low is reached before the SM time-out event, the channel switches to ON but the short / open detection is not started.

38.7.2 Monitoring the OFF State Width and the ADC Maximum Value

The modulation executed by the Reaction Channel can be monitored by measuring the width of the OFF states of the PWM waveforms provided by the channel. If the OFF state becomes too narrow it means that the load impedance is probably too high, thus indicating a possible open circuit. And if the sample value surpasses the `ADC_MAX_LIMIT` value a short circuit condition is detected. These criteria can be used for threshold-hold-on, and threshold low-period modulation modes.

The open detection event should occur twice in a row to set the OCDF flag. This condition helps to avoid false set of the open circuit detection flag. However a single sample that surpasses the `ADC_MAX_LIMIT` is sufficient to set the MAXL flag

The time limit for narrow OFF state is defined by `MIN_PWD` register. This value applies to all Reaction Channels. The PWM OFF states are measured by the Hold timers within the Reaction Channels. This is possible only during idle periods of this timer, for example from the moment a hold-on time-out occurs until the minimum threshold is reached when LOD is being used for the channel output pins. During these periods the hold counter is not used and thus it can measure the OFF states widths and compare them against the pre-defined limit defined by `MIN_PWD` register, as shown in the figure below.

Note

Consider an uncertainty of (+1) in the value `MIN_PWD` when calculating the OFF state width limit. The Hold timer prescaler contributes to this uncertainty..

For a programmed `MIN_PWD` value, a pulse wider than (`MIN_PWD+1`) is not an open circuit event to set the SCDF flag.

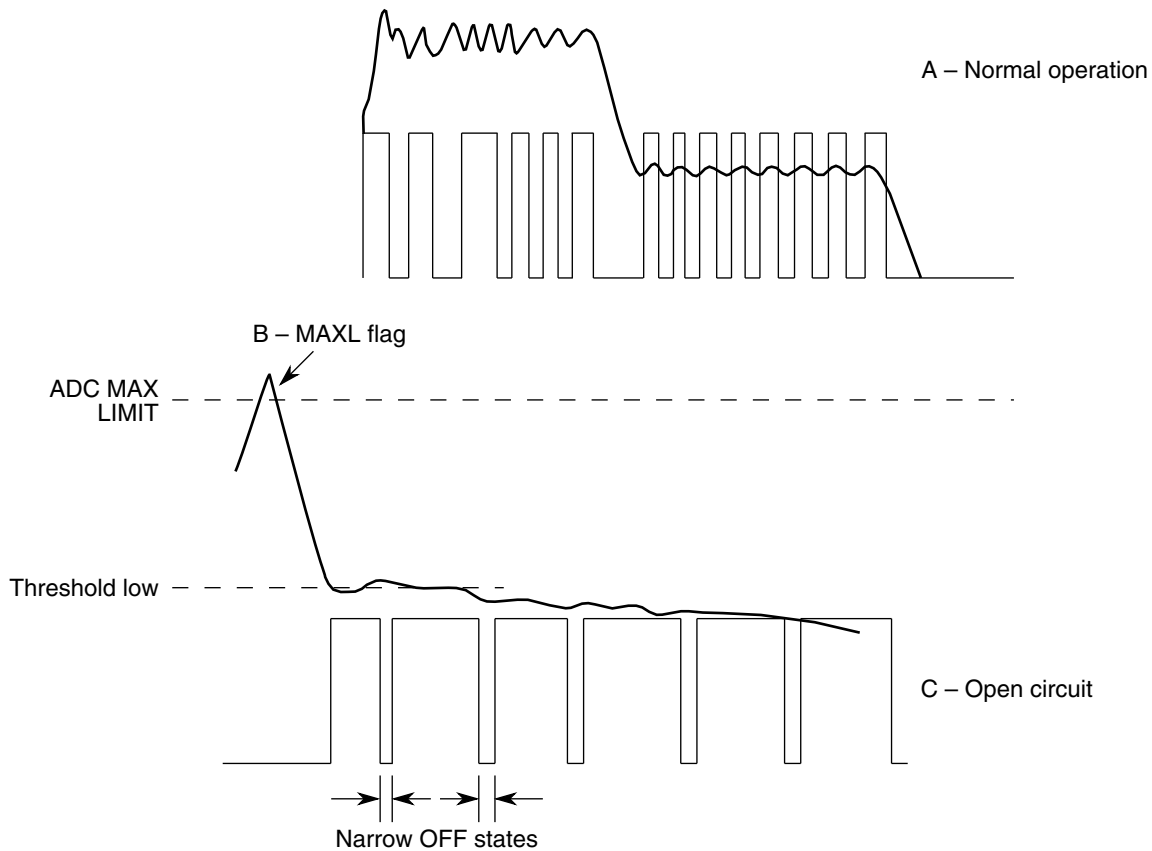


Figure 38-145. Fails detected by the OFF timing monitoring

The CHSR OCDF flag does not set if the OFF state was finished by disabling the modulation (i.e. eTPU channel=0 or SWMC=0) or by disabling the channel, CHEN=00, even if it ended shorter than MIN_PWD.

There can be a conflict of resource allocation if the Hold timer is used as the timer for the sequencer mode SM=10. In this case the RAER flag will be set to signal this conflict condition.

38.8 Reset Overview

The Reaction Module is reset whenever any MCU reset occurs. In order to re-initialize the Reaction Channels, the CHEN channel enable register should be used. Once disabled the channel output is set to DOFF state and the channel configuration can be changed safely. The disable/enable operation does not change the channel setup, thus the configuration registers remain at the state as before the channel was disabled. The Modulation Control word addressed by the channel after the enable bit is asserted is defined by the MODULATION ADDR field in the channel configuration register.

38.9 Reaction Module Interrupts

The Reaction Module issues one global interrupt signal and one interrupt signal per channel. If the Global interrupt signal is used, the resolution of the interrupt source needs to be performed by reading the Global Error Flag register to evaluate which channel issued the interrupt. After that the Channel Status register needs to be read to distinguish between the several interrupt sources by evaluating the flags: MAXL, OCDF, SCDF, TAER and SQER.

38.9.1 Interrupt sources

There are several sources of interrupts that indicates a faulty condition:

1. MAXL: maximum ADC result value was reached
2. OCDF: open circuit detected which indicates an open circuit, and thus, a potential malfunction in the circuitry controlled by the Reaction Module. Note that differently from the TAER, this flag does not indicate a faulty condition in the channel but in the circuit outside the SoC.
3. SCDF: short circuit detected which normally indicates a short circuit was detected on the off chip logic controlled by the reaction channel.
4. TAER: timer allocation error which indicates a required Timer resource was not allocated properly thus leading to faulty operation of the Reaction module.
5. SQER: sequencer error occurred meaning that the timer input signal was negated in a modulation phase with SM != 00.

38.9.2 Global Interrupt and Channel Interrupt

The following figure illustrates the relationship between the channel interrupts and the module global interrupt. The global interrupt summarizes the indication of error in the reaction module.

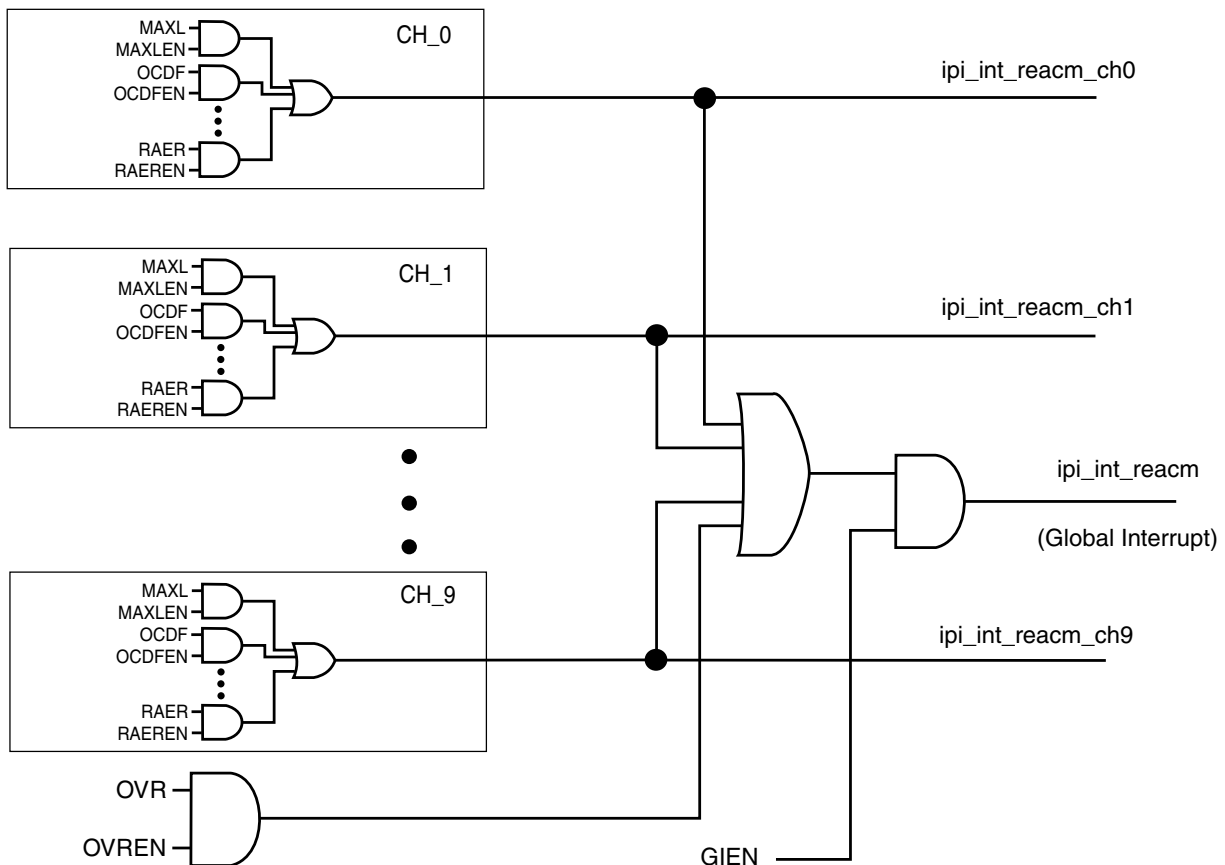


Figure 38-146. Channel Interrupt and Global Interrupt Diagram

38.10 Use cases

Figure 38-147 shows an example of the Reaction Module used to control an Injector solenoid. Note that this is a dual injector which is also called banked injector. Two Reaction channels are used to control this injector.

The injector Boost transistor on the top applies a higher voltage in order to minimize the time necessary for the injector to start injecting fuel. Transistors A and B control which injector is active in the injector bank. A sensor resistor is used to feed the current flowing through the solenoid back to the on-chip ADC. The current is sampled by the ADC and the result is sent to the Reaction Module, allowing closed loop control.

Note

This injector bank architecture does not allow both injectors to operate at the same time since the sensor in the feedback loop is shared by both injectors.

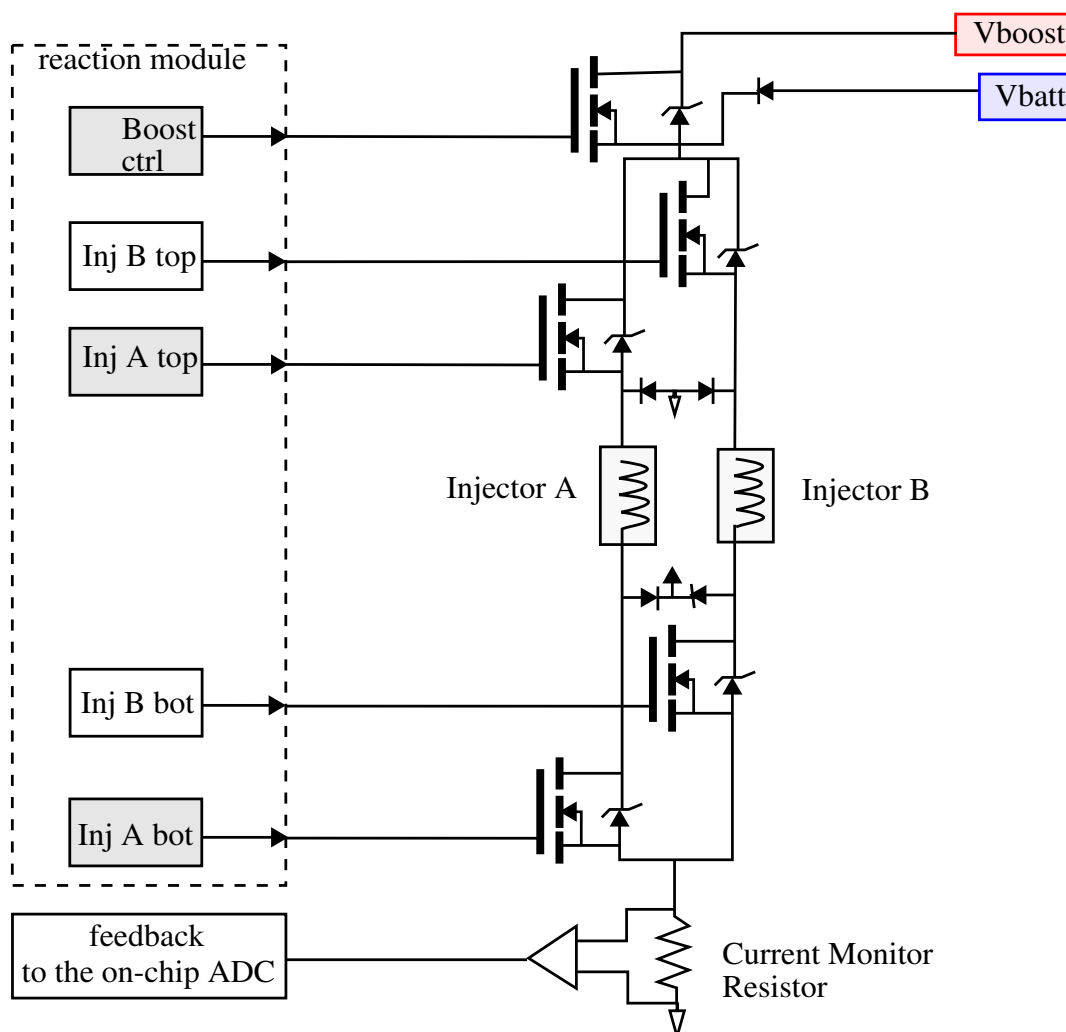


Figure 38-147. Boosted Banked Direct Injection with Passive Recirculation

Two eTPU channels are used to provide timing control signals, one for CH0 and one for CH1. The on-chip ADC monitors the sensor current periodically and sends the digitalized results to the Reaction Module. In a banked configuration as shown in this example one ADC channel is used to monitor both injectors current. Note that only one channel is *active* at a given time since the eTPU time windows are not *active* at the same time for both Reaction Channels. See [Figure 38-148](#) for more details.

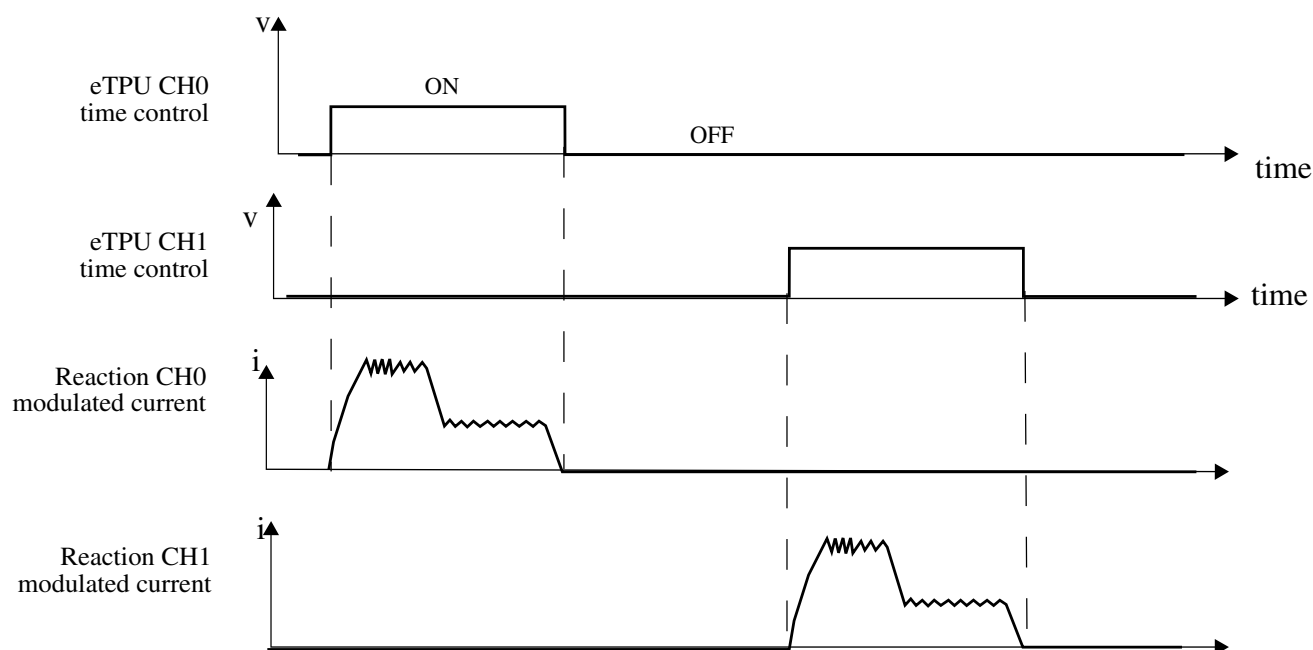


Figure 38-148. eTPU CH0/1 Controlling Reaction CH0/1

There are several possible configurations for a banked mode application. The objective on those configurations is to share hardware resources such as Reaction Module channels, ADC monitor inputs and MCU pins, among others related to the Injector driver which is not covered in detail in this document.

[Figure 38-149](#) shows a more detailed diagram of the interconnection between the injector bank and the Reaction Module. Two reaction channels are used in this application. CH0 is used to control Injector A and CH1 is used to control Injector B. However, the Vboost/Vbatt selection is controlled by CH0 ch0_a output only since, when VBoost driver is switched off, Vbatt power source is applied to the injectors by the direct bias of the diode connecting Vbatt to the injector bank.

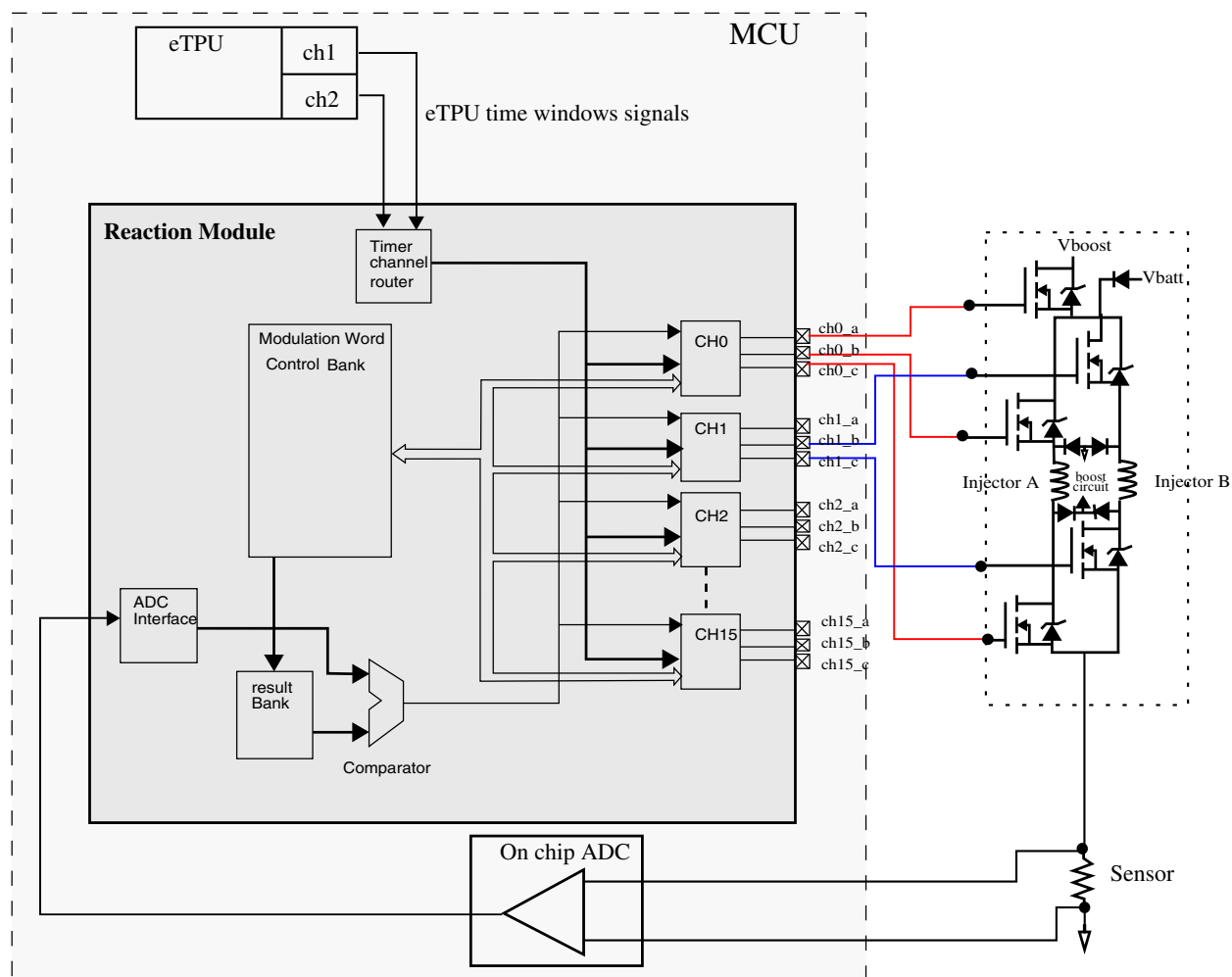


Figure 38-149. System Level Connection in a Banked Configuration

It is important to note that even if two reaction channels control different injectors they can share the data stored in the Modulation Word Control. In this case both channels should execute the same type of modulation and use the same threshold values. Note also that the data stored in the Threshold bank in this case is also shared between these channels. This is an important feature of the Reaction Module architecture since it allows the sharing of resources and therefore provides savings in size without compromising the module functionality.

Figure 38-150 shows an example of the required current levels through Injector A and B. In order to generate this waveform, the Reaction Module uses one Modulation Control Word for each one of the four phases of the waveform from A through D. In this example the Module should be configured in the following way:

1. Set the REACM2_CHRR0 CHIR[3:0]=0x0, thus routing eTPU channel 0 to Reaction Channel 0
2. Set the REACM2_CHRR1 CHIR[3:0]=0x1, thus routing eTPU channel 1 to Reaction Channel 1
3. Set the REACM2_CHRR0 ADCR[3:0]=0x0, thus routing ADC TAG 0 to Reaction Channel 0
4. Set the REACM2_CHRR1 ADCR[3:0]=0x0, thus routing ADC TAG 0 to Reaction Channel 1
5. Program Modulation Word Control bank according to [Figure 38-151](#)
6. Program Shared Timer Bank REACM2_STBK for addresses from 0 through 1 with timing intervals related to the duration of phases B and C respectively.
7. Program appropriate values in the Threshold Bank. Since threshold-threshold modulation is to be used in this example, 4 pairs of values should be provided for phases A,B,C and D respectively. Each pair corresponds to one address of the REACM2_THBK starting at address 0x0400.
8. Program configuration registers for both channels, REACM2_CHCR0/1. The parameters are DOFF[2:0] which defines the OFF state of the channel outputs and the MODULATION_ADDR = 0x0, which defines the address of the Modulation Control word. It is assumed that the Modulation Word zero is the first word to be accessed by both channels. Since 4 Modulation words will be used the addresses will be incremented by the Reaction Channel as needed, thus only the address for the first word is required. Note that MODULATION_ADDR=0x0 points to the first Modulation Word in the Modulation Word Bank.
9. Program the prescalers HPRE and TPRES in the REACM2_TCR register. Also enable the prescalers by setting the TPRES and HPRES bits in the REACM2_MCR register.
10. Enable channels CH0 and CH1 to start the modulation sequence by programming field CHEN=01 on REACM2_CHCR1/0 registers. At this time the Reaction channel CH0 accesses the Modulation Control word zero and switches to ON state as defined by the data stored in the HOD[2:0] field. Up to this point any activity in the eTPU channel or incoming ADC result is ignored by the Reaction module. After CHEN field is programmed, the reaction channels wait until a timer window is initiated by eTPU for the modulation process to start.

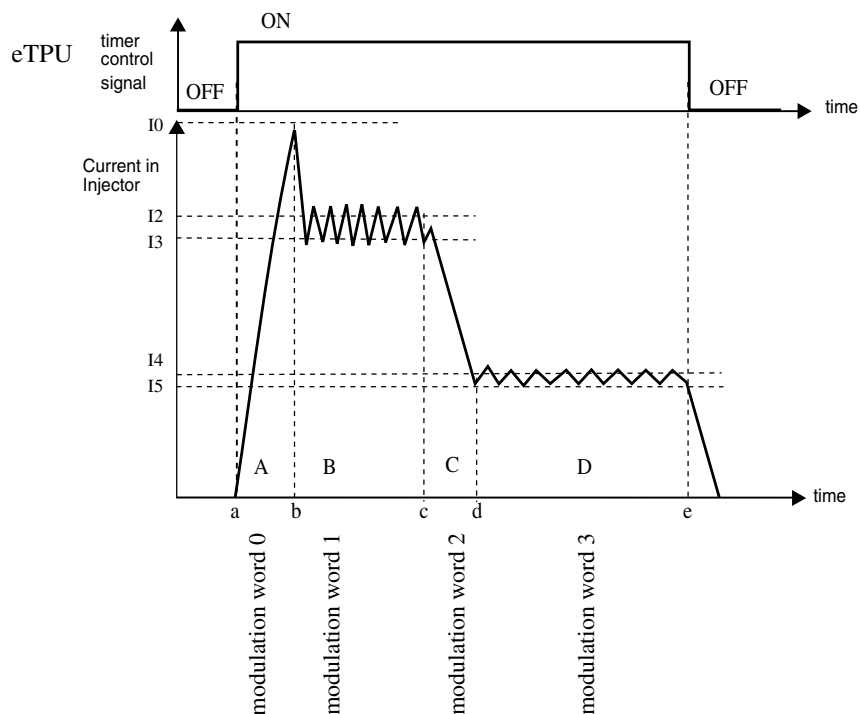


Figure 38-150. Modulation Phases1

The current through the injectors are initially zero since the reaction channel output is in the OFF state as configured by the DOFF[2:0] field. The eTPU timer window is at the OFF state as well. The modulation starts when the eTPU channel time window switches to ON state. Modulation *phase A* starts at this time as described in [Figure 38-150](#). The modulation word 0 is used. In this application this phase corresponds to the *Boosted Peak phase*. This process allows the fuel injection to start sooner because of the sharp edge of the current which is important for a precise control of the fuel to be injected. *Phase A* is setup to execute a Threshold-Threshold modulation but note that only *I0* value is used since the phase advances when a certain current is achieved. Below is a description of the bit fields in the Modulation word in order to execute the modulation described in *phase A*:

- LOOP=0
- The initial value should be HOD (IOSS=1)
- Threshold-Threshold modulation mode is obtained with MM=00
- This phase ends when the threshold value *I0* is achieved (SM=11)
- Necessary to have HOD=111 for boosted operation and sensor active. LOD setting is not important in this case since it is not used
- *I0* is read from Threshold Bank by using THRESPT=0x0 that points to address 0 of this bank

Use cases

- *I1* is read from Threshold Bank by using $(\text{THRESPT}+1) = 0x1$
- Hold timer is not used, therefore *HDOFFPT* can have any value (X).

At point b, *phase B* is initiated by the threshold being achieved from *phase A*. CH0 increments the Modulation Word address to *MODULATION_ADDR* = 0x1 and the second Modulation Word is read by the channel. As a result of the second Modulation Word decoding the *Vboost* voltage is disabled causing a *peak* modulation with *Vbatt*. The phase is called the *Peak Vbatt phase*. For *phase B* a Threshold-Threshold modulation is used with levels *I2* and *I3* during a period defined by *TB*. See [Figure 38-151](#)

A time-out event is received from the Shared Timer sub-module and a new Modulation Word is read. *Phase C* corresponds to the *re-circulation phase*. Energy from the injector is transferred back to the boost circuitry. In this phase the current cannot be measured because there is no current flowing through the sensor resistor. A Threshold-Threshold modulation mode is used. The Shared Timer is started at the beginning of this phase and *TC* delay is measured. The channel outputs are kept in the OFF state.

When the Shared Timer times out after *TC* delay the Modulation Word address is incremented, *MODULATION_ADDR* = 0x4 and the *Hold phase* is initiated. This phase is typically longer compared to the other phases and defines the amount of fuel that will be injected. Threshold-Threshold modulation mode is used between levels *I4* and *I5* and *Vbatt* is selected as the power supply. The phase ends based on the eTPU time window switching to *off* at point e. At this time the channel outputs are set to OFF and the channel points to *MODULATION_ADDR* = 0x0 which is the address of the first Modulation Word. Note that this address is not necessarily 0x0.

This modulation process is executed by a sequence of Modulation Words as described in [Figure 38-151](#).

Modulation Control Word Bank (REACM2_MWBKn)																																		
	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0		
	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31		
Fields	L	I			MM			SM			HOD				LOD			THRESHPT							STPT						HDOFFPT			
Phase A	0	1			00			11			111				101			000000 (I0) 000001 (I1)							X						X			
Phase B	0	0			00			01			110				100			000010 (I2) 000011 (I3)							000000 (TB)						X			
Phase C	0	0			00			01			000				000			X ¹							0001 (TC)						X			
Phase D	0	0			00			00			110				100			000100 (I4) 000101 (I5)							X						X			

Figure 38-151. Modulation Words for Injector Application

Note

¹ Any value

38.10.1 Advancing Modulation Phase on a Threshold level

The Modulation Phase may be set to advance when a specific threshold value is reached. The use case for this scenario is described in the following figure. This functionality is used to assure a specific current level was achieved before the Reaction Channel advances to the next modulation phase, thus making sure that the solenoid had the fastest opening speed.

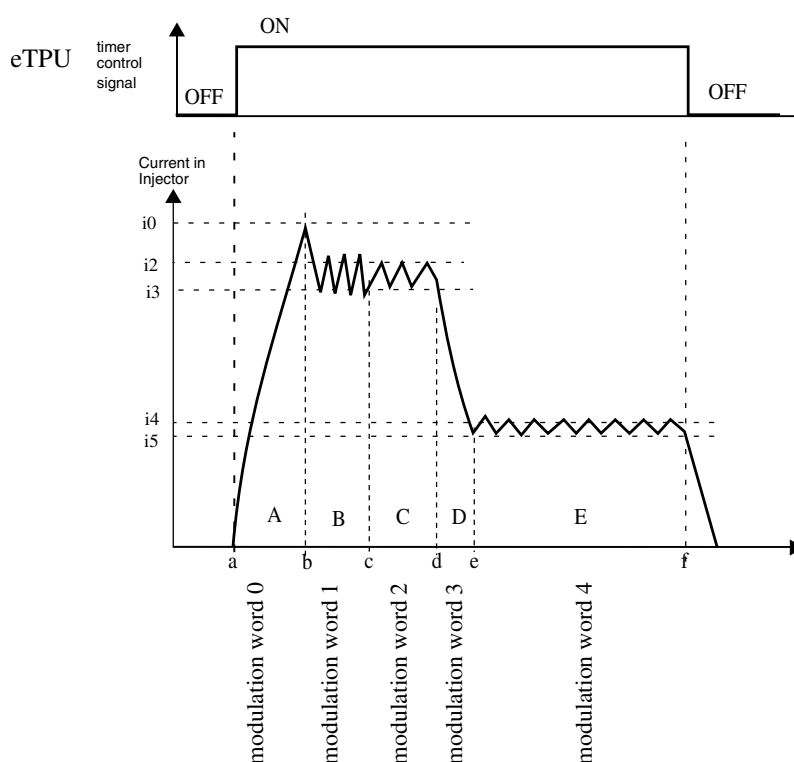


Figure 38-152. Advancing Modulation Phase on a Threshold level

In the example shown in the preceding figure the Modulation Word 0 has the field SM=11 meaning that it should advance if the ADC comparator, which compares ADC results and threshold values, is true. This means that the ADC result is greater or equal to the threshold value pointed by THRESPT Modulation Word 0 field. The advance from phase A to phase B in the preceding figure is described as follows:

- Initially at time *a* the Reaction Channel output is loaded with HOD[2:0] as indicated by IOSS value, both fields of modulation word 0.
- If ADC result $\geq$ [THRESPT] the Reaction Channel turns the outputs off by loading LOD[2:0] to the channel outputs and advance to the next modulation cycle at time *b*.

Note

The advance on threshold, SM=11, is intended to be used with IOSS=1, thus the advance occurs when the level from the ADC is greater or equal to the value pointed by THRESPT. IOSS=0 is a reserved value for this bit in this configuration and should not be used.

Note

In case of MM=01 (threshold/holdoff) on phase B, a glitch can occur in the beginning of this phase. To avoid this, on phase A it is necessary to set HOD=LOD. However, a false flag SCDF can be set at this time and should be discarded.

38.10.2 Controlling the LOOP function

The LOOP field in the Modulation Word controls the sequencing of Modulation Words to be executed by the channel. If LOOP = 1 and a phase is ended the next Modulation Word address returns to the initial modulation address programmed in the channel when the cycle was initiated. In this case a loop in the Modulation Bank is created. This loop ends when the cycle ends.

The following figure shows the sequence of Modulation Words within a Modulation Cycle if LOOP function is used. Note that an alternate modulated waveform is generated using only two Modulation Words, in this case Modulation Word 0 and Modulation Word 1.

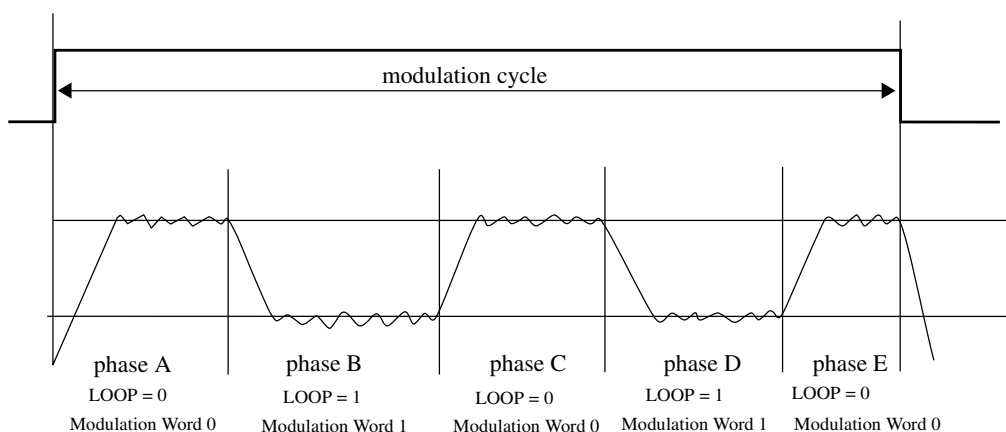


Figure 38-153. LOOP function used within a Modulation Cycle

NOTE

Due to the limit of 16 words for a channel modulation, after the 16th modulation word accesses, the sequence advance is executed by returning back to the first word of the sequence even with LOOP=0.

38.10.3 Banked Mode

The following figure describes the interconnection of four channels controlling two injector banks. Note that two output pins are not connected since the boost control is shared between channels [0] and [1] and channels [2] and [3].

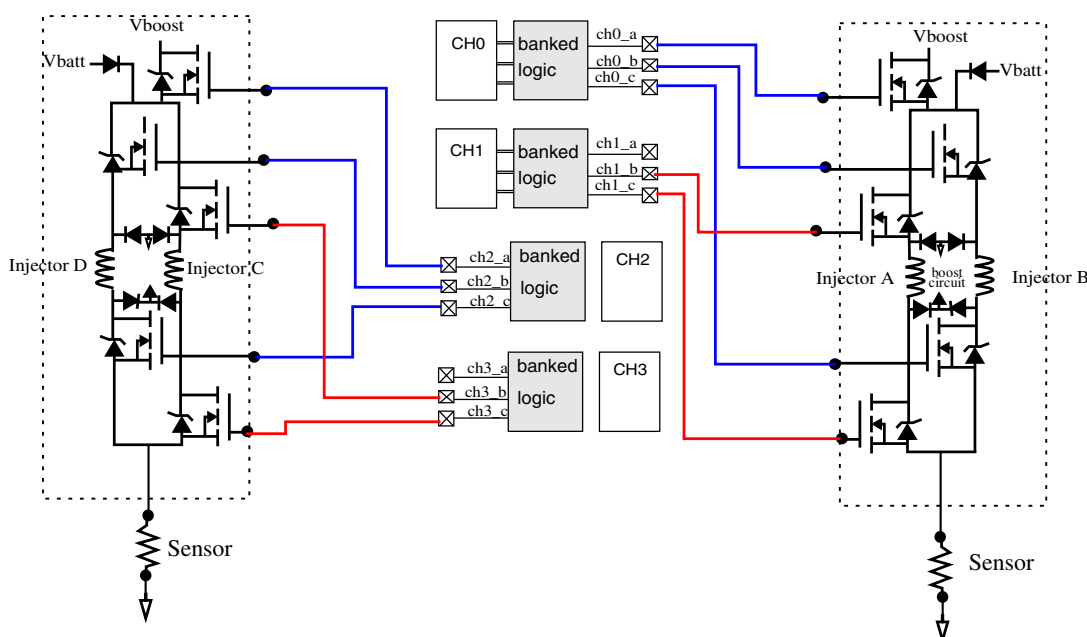


Figure 38-154. Four channels controlling two injector banks in banked mode.

Chapter 39

Input Glitch Filter (IGF)

39.1 Chip-specific IGF information

This chip has two instances of the IGF module: IGF_0 and IGF_1. They are in the eTPU clock domain. The two instances are located on different Peripheral Bridges.

39.1.1 IGF channels and internal prescalers

The two IGF module instances support a combined total of 76 channels. Some channels of the instances implement internal prescalers.

Each channel has a 24-bit counter. Each internal prescaler is 9 bits wide.

The following table identifies the number of channels of each instance, the specific channel numbers that implement an internal prescaler, and the group of specific channel numbers that can use each implemented internal prescaler.

Each group of channel numbers that is associated with one implemented prescaler is a filter cluster.

Table 39-1. IGF instance channels and internal prescalers

Instance	Channels supported	Channel number with prescaler	Filter cluster: channel numbers that use prescaler
IGF_0	30	0	0-29
IGF_1	46	0	0-31
		32	32-45

39.1.2 IGF internal prescaler selection

Channels in a filter cluster that do not implement their own internal prescaler must select an "external" prescaler. This selection means the channel uses the prescaler of the preceding channel in the filter cluster.

Within a filter cluster, to enable a channel x to use the cluster's internal prescaler:

1. For the channel that implements the cluster's internal prescaler, select the internal prescaler by ensuring that the channel's $\text{MCR}_n[\text{PSSEL}]$ field is 0.
2. For every channel between that channel and channel x , including channel x itself, select the "external" prescaler by setting the channel's $\text{MCR}_n[\text{PSSEL}]$ field to 1.

39.1.2.1 Example of internal prescaler selection

To select the applicable internal prescaler for IGF_0 channel 3:

1. Because IGF_0's channel 0 implements the cluster's internal prescaler, select the internal prescaler for channel 0 by ensuring that IGF_0's $\text{MCR}_0[\text{PSSEL}]$ field is 0.
2. Select the "external" prescaler for every channel between channel 0 and channel 3, including channel 3 itself:
 - a. Set IGF_0's $\text{MCR}_1[\text{PSSEL}]$ to 1.
 - b. Set IGF_0's $\text{MCR}_2[\text{PSSEL}]$ to 1.
 - c. Set IGF_0's $\text{MCR}_3[\text{PSSEL}]$ to 1.

39.1.2.2 Restriction on prescaler selection for IGF_1 channel 32

Do not set IGF_1's $\text{MCR}_{32}[\text{PSSEL}]$ field to 1.

This setting could cause timing failure for all IGF_1 channels.

39.1.3 IGF interaction with other modules

The IGF modules interface with eTPU and eMIOS modules.

The following table shows the mapping of IGF_0 channels.

Table 39-2. IGF_0 channel mapping

Filter channel	Primary function pin	Module
0	TCRCLKC	eTPU_C
1-15	ETPUC0-14	eTPU_C
16-19	EMIOS0_0-3 ¹	eMIOS_0
20-21	EMIOS0_6-7	eMIOS_0
22-29	EMIOS0_16-23	eMIOS_0

1. EMIOS0_4 and EMIOS0_5 do not have filters because the input channels are not connected directly from pads. They are connected from the DSPI_B module.

The following table shows the mapping of IGF_1 channels.

Table 39-3. IGF_1 channel mapping

Filter channel	Primary function pin	Module
0	TCRCLKA	eTPU_A
1-15	ETPUA0-14	eTPU_A
16	TCRCLKB	eTPU_B
17-31	ETPUB0-14	eTPU_B
32-37	EMIOS1_0-5 ¹	eMIOS_1
38-45	EMIOS1_16-23	eMIOS_1

1. EMIOS1_6 and EMIOS1_7 do not have filters because the input channels are not connected directly from pads. They are connected from the DSPI_D module.

39.1.4 STOP mode system requests for IGF_0 and IGF_1

As described in [STOP mode](#), a STOP mode system request can stop the IGF module instances.

This chip also has inter-module STOP mode system requests that affect IGF_0 and IGF_1 individually:

- IGF_0 automatically stops when eTPU_C and eMIOS_0 stop.
- IGF_1 automatically stops when eTPU_A/eTPU_B and eMIOS_1 stop.

39.2 Introduction

The Input Glitch Filter provides programmable glitch rejection filtering of digital input signals.

39.3 Block Diagram

The figure below is the block diagram for the Input Glitch Filter module.

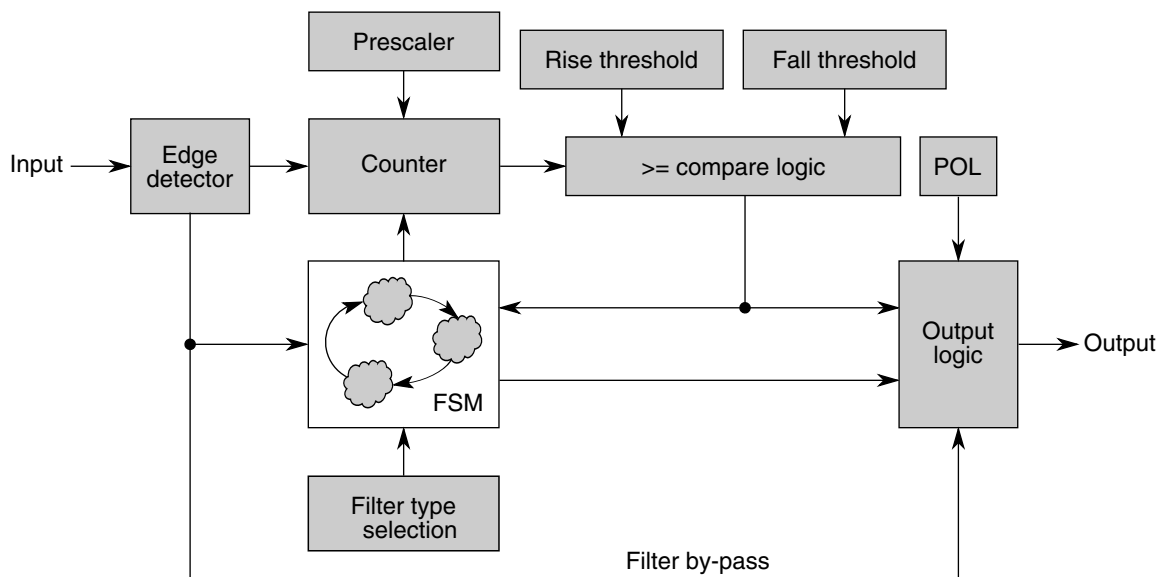


Figure 39-1. Input Glitch Filter block diagram

39.4 Input Glitch Filter overview

The Input Glitch Filter receives an input signal usually from a device PAD. Signals generated externally to the device are affected by environmental noise or glitches. The signal level has to be evaluated with some criteria in order to remove the noise which could affect the measurements done over this signal, such as transition time, period and level.

In a typical application, the Input Glitch Filter interfaces with one or more timer peripherals.

The strategy to remove glitches from an input signal is to monitor the level of the signal during a wide time window thus the glitches can be evaluated and removed.

The input signal should be stable for a certain amount of time before it can be used as a trigger or by a timer input channel. This strategy implies a delay on detecting a transition in the signal.

There is another strategy which is based on removing the glitches by introducing a blanking period after an edge is detected. In this strategy there is no delay from the edge detection in the input signal to the filtered signal, in the filter output. The strategy also allows input signals with different filter periods to be measured at the same time.

39.4.1 Interaction with other peripherals

The figure below shows an example of how the Input Glitch Filter interacts with on-chip peripherals.

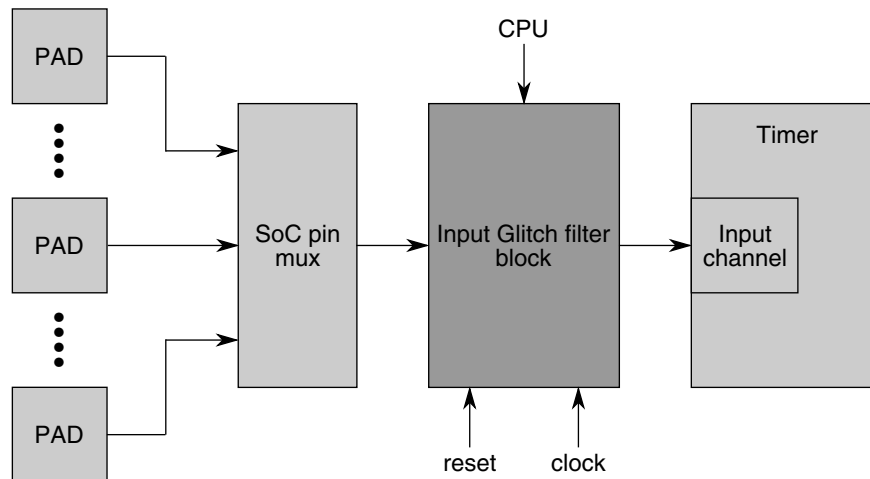


Figure 39-2. Input Glitch Filter application example

39.4.2 Mode of operation

The Input Glitch Filter has the following modes of operation:

- Configuration mode
- Normal mode
- Debug mode
- Low power mode

39.4.2.1 Configuration mode

The configuration mode is the initial mode of the Input Glitch filter after system reset is applied. The configuration mode is selected when FGEN = '0' in the MCR register. The filter parameters such as rising and falling thresholds can be programmed when the filter

is in configuration mode without disturbing the internal device logic since the filter output is kept at logic '0' state. Once FGEN is programmed with logic '1' the filter enters the normal operating mode and is able to initiate the filtering process.

NOTE

The MDIS bit in the MCR register, if set to logic '1', makes the filter transition to low power mode independent of FGEN bit settings. Thus, in order to enter the configuration mode MDIS bit needs to be at logic '0'.

39.4.2.2 Normal mode

The normal mode of operation is entered when MCR register FGEN = '1'. In this mode the filter implements one of the filter types:

- Bypass filter
- Windowing filter
- Integrating filter
- Integrating-hold filter

These filter types are described in the following sections. Before entering the normal mode, the filter parameters, such as rise and fall thresholds, need to be programmed in the appropriate registers, as well as the type needs to be defined in the corresponding register.

NOTE

In the normal mode it is still possible to change the filter parameters, such as rise and fall thresholds. See section [On-the-fly filter reconfiguration](#) for more details.

39.4.2.3 Debug mode

The debug mode is entered when the MCR FRZ bit is set and a system level debug request occurs. Once the debug mode is entered, the internal filter counter is halted and the debug functions controlled by the FBP, FOH, and FOL bits are enabled. The filter output is frozen and remains at the state it was immediately before the debug mode was entered. The FOH and FOL MCR bits can be used to force the filter output to logic '1' or '0', respectively. These bits are self-negating, thus they always read as zero.

By setting the MCR FBP bit, the filter input state is copied to the filter output, thus allowing to control the filter output based on the filter input state. The output is forced during one clock cycle. After that the filter output remains at the new forced state until it is changed again by using a debug control bit or if the filter returns to the normal mode of operation.

NOTE

The system level debug request occurs when a specific event, or a series of events, occurs in one of the device sub-modules and is interpreted by the device debug logic as a debug request. The Input Glitch Filter is not capable of generating an event that causes the assertion of the system level debug request.

39.4.2.4 Low power mode

The low power mode is controlled by the MDIS bit in the MCR register or by the STOP signal which is the system level low power control signal. When entering low power mode the filter output remains unchanged at the state it was just before the low power mode was entered. Since the Input Glitch Filter clocks are stopped in this mode, the internal counter remains inactive. The description of this mode is covered in the Low Power Mode section of this document.

39.5 Memory map

The Input Glitch Filter registers are implemented in 32-bit. 8-bit write operations are allowed. The coherency in case of 8-bit access must be provided by the application software since there is no coherence mechanism built into the hardware.

NOTE

RESERVED bit registers, when available for write operations, should be written always with logic '0'. Read access does not return meaningful data from RESERVED bit registers.

IGF memory map

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
0	Module Configuration Register (IGF_MCR0)	32	R/W	0000_0000h	39.5.1/1924
4	Module Status Register (IGF_MSR0)	32	R/W	0000_0000h	39.5.2/1927

Table continues on the next page...

IGF memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
10	Prescaler Register (IGF_PRESR0)	32	R/W	0000_0000h	39.5.3/1928
1C	Rising edge Threshold Register (IGF_RTHR0)	32	R/W	0000_0000h	39.5.4/1929
20	Falling edge Threshold Register (IGF_FTHR)	32	R/W	0000_0000h	39.5.5/1929
40	Module Configuration Register (IGF_MCR1)	32	R/W	0000_0000h	39.5.1/1924
44	Module Status Register (IGF_MSR1)	32	R/W	0000_0000h	39.5.2/1927
5C	Rising edge Threshold Register (IGF_RTHR1)	32	R/W	0000_0000h	39.5.4/1929
80	Module Configuration Register (IGF_MCR2)	32	R/W	0000_0000h	39.5.1/1924
84	Module Status Register (IGF_MSR2)	32	R/W	0000_0000h	39.5.2/1927
9C	Rising edge Threshold Register (IGF_RTHR2)	32	R/W	0000_0000h	39.5.4/1929
C0	Module Configuration Register (IGF_MCR3)	32	R/W	0000_0000h	39.5.1/1924
C4	Module Status Register (IGF_MSR3)	32	R/W	0000_0000h	39.5.2/1927
DC	Rising edge Threshold Register (IGF_RTHR3)	32	R/W	0000_0000h	39.5.4/1929
100	Module Configuration Register (IGF_MCR4)	32	R/W	0000_0000h	39.5.1/1924
104	Module Status Register (IGF_MSR4)	32	R/W	0000_0000h	39.5.2/1927
11C	Rising edge Threshold Register (IGF_RTHR4)	32	R/W	0000_0000h	39.5.4/1929
140	Module Configuration Register (IGF_MCR5)	32	R/W	0000_0000h	39.5.1/1924
144	Module Status Register (IGF_MSR5)	32	R/W	0000_0000h	39.5.2/1927
15C	Rising edge Threshold Register (IGF_RTHR5)	32	R/W	0000_0000h	39.5.4/1929
180	Module Configuration Register (IGF_MCR6)	32	R/W	0000_0000h	39.5.1/1924
184	Module Status Register (IGF_MSR6)	32	R/W	0000_0000h	39.5.2/1927
19C	Rising edge Threshold Register (IGF_RTHR6)	32	R/W	0000_0000h	39.5.4/1929
1C0	Module Configuration Register (IGF_MCR7)	32	R/W	0000_0000h	39.5.1/1924
1C4	Module Status Register (IGF_MSR7)	32	R/W	0000_0000h	39.5.2/1927
1DC	Rising edge Threshold Register (IGF_RTHR7)	32	R/W	0000_0000h	39.5.4/1929
200	Module Configuration Register (IGF_MCR8)	32	R/W	0000_0000h	39.5.1/1924
204	Module Status Register (IGF_MSR8)	32	R/W	0000_0000h	39.5.2/1927
21C	Rising edge Threshold Register (IGF_RTHR8)	32	R/W	0000_0000h	39.5.4/1929
240	Module Configuration Register (IGF_MCR9)	32	R/W	0000_0000h	39.5.1/1924
244	Module Status Register (IGF_MSR9)	32	R/W	0000_0000h	39.5.2/1927
25C	Rising edge Threshold Register (IGF_RTHR9)	32	R/W	0000_0000h	39.5.4/1929
280	Module Configuration Register (IGF_MCR10)	32	R/W	0000_0000h	39.5.1/1924
284	Module Status Register (IGF_MSR10)	32	R/W	0000_0000h	39.5.2/1927
29C	Rising edge Threshold Register (IGF_RTHR10)	32	R/W	0000_0000h	39.5.4/1929
2C0	Module Configuration Register (IGF_MCR11)	32	R/W	0000_0000h	39.5.1/1924
2C4	Module Status Register (IGF_MSR11)	32	R/W	0000_0000h	39.5.2/1927
2DC	Rising edge Threshold Register (IGF_RTHR11)	32	R/W	0000_0000h	39.5.4/1929
300	Module Configuration Register (IGF_MCR12)	32	R/W	0000_0000h	39.5.1/1924
304	Module Status Register (IGF_MSR12)	32	R/W	0000_0000h	39.5.2/1927
31C	Rising edge Threshold Register (IGF_RTHR12)	32	R/W	0000_0000h	39.5.4/1929

Table continues on the next page...

IGF memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
340	Module Configuration Register (IGF_MCR13)	32	R/W	0000_0000h	39.5.1/1924
344	Module Status Register (IGF_MSR13)	32	R/W	0000_0000h	39.5.2/1927
35C	Rising edge Threshold Register (IGF_RTHR13)	32	R/W	0000_0000h	39.5.4/1929
380	Module Configuration Register (IGF_MCR14)	32	R/W	0000_0000h	39.5.1/1924
384	Module Status Register (IGF_MSR14)	32	R/W	0000_0000h	39.5.2/1927
39C	Rising edge Threshold Register (IGF_RTHR14)	32	R/W	0000_0000h	39.5.4/1929
3C0	Module Configuration Register (IGF_MCR15)	32	R/W	0000_0000h	39.5.1/1924
3C4	Module Status Register (IGF_MSR15)	32	R/W	0000_0000h	39.5.2/1927
3DC	Rising edge Threshold Register (IGF_RTHR15)	32	R/W	0000_0000h	39.5.4/1929
400	Module Configuration Register (IGF_MCR16)	32	R/W	0000_0000h	39.5.1/1924
404	Module Status Register (IGF_MSR16)	32	R/W	0000_0000h	39.5.2/1927
41C	Rising edge Threshold Register (IGF_RTHR16)	32	R/W	0000_0000h	39.5.4/1929
440	Module Configuration Register (IGF_MCR17)	32	R/W	0000_0000h	39.5.1/1924
444	Module Status Register (IGF_MSR17)	32	R/W	0000_0000h	39.5.2/1927
45C	Rising edge Threshold Register (IGF_RTHR17)	32	R/W	0000_0000h	39.5.4/1929
480	Module Configuration Register (IGF_MCR18)	32	R/W	0000_0000h	39.5.1/1924
484	Module Status Register (IGF_MSR18)	32	R/W	0000_0000h	39.5.2/1927
49C	Rising edge Threshold Register (IGF_RTHR18)	32	R/W	0000_0000h	39.5.4/1929
4C0	Module Configuration Register (IGF_MCR19)	32	R/W	0000_0000h	39.5.1/1924
4C4	Module Status Register (IGF_MSR19)	32	R/W	0000_0000h	39.5.2/1927
4DC	Rising edge Threshold Register (IGF_RTHR19)	32	R/W	0000_0000h	39.5.4/1929
500	Module Configuration Register (IGF_MCR20)	32	R/W	0000_0000h	39.5.1/1924
504	Module Status Register (IGF_MSR20)	32	R/W	0000_0000h	39.5.2/1927
51C	Rising edge Threshold Register (IGF_RTHR20)	32	R/W	0000_0000h	39.5.4/1929
540	Module Configuration Register (IGF_MCR21)	32	R/W	0000_0000h	39.5.1/1924
544	Module Status Register (IGF_MSR21)	32	R/W	0000_0000h	39.5.2/1927
55C	Rising edge Threshold Register (IGF_RTHR21)	32	R/W	0000_0000h	39.5.4/1929
580	Module Configuration Register (IGF_MCR22)	32	R/W	0000_0000h	39.5.1/1924
584	Module Status Register (IGF_MSR22)	32	R/W	0000_0000h	39.5.2/1927
59C	Rising edge Threshold Register (IGF_RTHR22)	32	R/W	0000_0000h	39.5.4/1929
5C0	Module Configuration Register (IGF_MCR23)	32	R/W	0000_0000h	39.5.1/1924
5C4	Module Status Register (IGF_MSR23)	32	R/W	0000_0000h	39.5.2/1927
5DC	Rising edge Threshold Register (IGF_RTHR23)	32	R/W	0000_0000h	39.5.4/1929
600	Module Configuration Register (IGF_MCR24)	32	R/W	0000_0000h	39.5.1/1924
604	Module Status Register (IGF_MSR24)	32	R/W	0000_0000h	39.5.2/1927
61C	Rising edge Threshold Register (IGF_RTHR24)	32	R/W	0000_0000h	39.5.4/1929
640	Module Configuration Register (IGF_MCR25)	32	R/W	0000_0000h	39.5.1/1924
644	Module Status Register (IGF_MSR25)	32	R/W	0000_0000h	39.5.2/1927
65C	Rising edge Threshold Register (IGF_RTHR25)	32	R/W	0000_0000h	39.5.4/1929

Table continues on the next page...

IGF memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
680	Module Configuration Register (IGF_MCR26)	32	R/W	0000_0000h	39.5.1/1924
684	Module Status Register (IGF_MSR26)	32	R/W	0000_0000h	39.5.2/1927
69C	Rising edge Threshold Register (IGF_RTHR26)	32	R/W	0000_0000h	39.5.4/1929
6C0	Module Configuration Register (IGF_MCR27)	32	R/W	0000_0000h	39.5.1/1924
6C4	Module Status Register (IGF_MSR27)	32	R/W	0000_0000h	39.5.2/1927
6DC	Rising edge Threshold Register (IGF_RTHR27)	32	R/W	0000_0000h	39.5.4/1929
700	Module Configuration Register (IGF_MCR28)	32	R/W	0000_0000h	39.5.1/1924
704	Module Status Register (IGF_MSR28)	32	R/W	0000_0000h	39.5.2/1927
71C	Rising edge Threshold Register (IGF_RTHR28)	32	R/W	0000_0000h	39.5.4/1929
740	Module Configuration Register (IGF_MCR29)	32	R/W	0000_0000h	39.5.1/1924
744	Module Status Register (IGF_MSR29)	32	R/W	0000_0000h	39.5.2/1927
75C	Rising edge Threshold Register (IGF_RTHR29)	32	R/W	0000_0000h	39.5.4/1929
780	Module Configuration Register (IGF_MCR30)	32	R/W	0000_0000h	39.5.1/1924
784	Module Status Register (IGF_MSR30)	32	R/W	0000_0000h	39.5.2/1927
79C	Rising edge Threshold Register (IGF_RTHR30)	32	R/W	0000_0000h	39.5.4/1929
7C0	Module Configuration Register (IGF_MCR31)	32	R/W	0000_0000h	39.5.1/1924
7C4	Module Status Register (IGF_MSR31)	32	R/W	0000_0000h	39.5.2/1927
7DC	Rising edge Threshold Register (IGF_RTHR31)	32	R/W	0000_0000h	39.5.4/1929
800	Module Configuration Register (IGF_MCR32)	32	R/W	0000_0000h	39.5.1/1924
804	Module Status Register (IGF_MSR32)	32	R/W	0000_0000h	39.5.2/1927
810	Prescaler Register (IGF_PRESR32)	32	R/W	0000_0000h	39.5.3/1928
81C	Rising edge Threshold Register (IGF_RTHR32)	32	R/W	0000_0000h	39.5.4/1929
840	Module Configuration Register (IGF_MCR33)	32	R/W	0000_0000h	39.5.1/1924
844	Module Status Register (IGF_MSR33)	32	R/W	0000_0000h	39.5.2/1927
85C	Rising edge Threshold Register (IGF_RTHR33)	32	R/W	0000_0000h	39.5.4/1929
880	Module Configuration Register (IGF_MCR34)	32	R/W	0000_0000h	39.5.1/1924
884	Module Status Register (IGF_MSR34)	32	R/W	0000_0000h	39.5.2/1927
89C	Rising edge Threshold Register (IGF_RTHR34)	32	R/W	0000_0000h	39.5.4/1929
8C0	Module Configuration Register (IGF_MCR35)	32	R/W	0000_0000h	39.5.1/1924
8C4	Module Status Register (IGF_MSR35)	32	R/W	0000_0000h	39.5.2/1927
8DC	Rising edge Threshold Register (IGF_RTHR35)	32	R/W	0000_0000h	39.5.4/1929
900	Module Configuration Register (IGF_MCR36)	32	R/W	0000_0000h	39.5.1/1924
904	Module Status Register (IGF_MSR36)	32	R/W	0000_0000h	39.5.2/1927
91C	Rising edge Threshold Register (IGF_RTHR36)	32	R/W	0000_0000h	39.5.4/1929
940	Module Configuration Register (IGF_MCR37)	32	R/W	0000_0000h	39.5.1/1924
944	Module Status Register (IGF_MSR37)	32	R/W	0000_0000h	39.5.2/1927
95C	Rising edge Threshold Register (IGF_RTHR37)	32	R/W	0000_0000h	39.5.4/1929
980	Module Configuration Register (IGF_MCR38)	32	R/W	0000_0000h	39.5.1/1924
984	Module Status Register (IGF_MSR38)	32	R/W	0000_0000h	39.5.2/1927

Table continues on the next page...

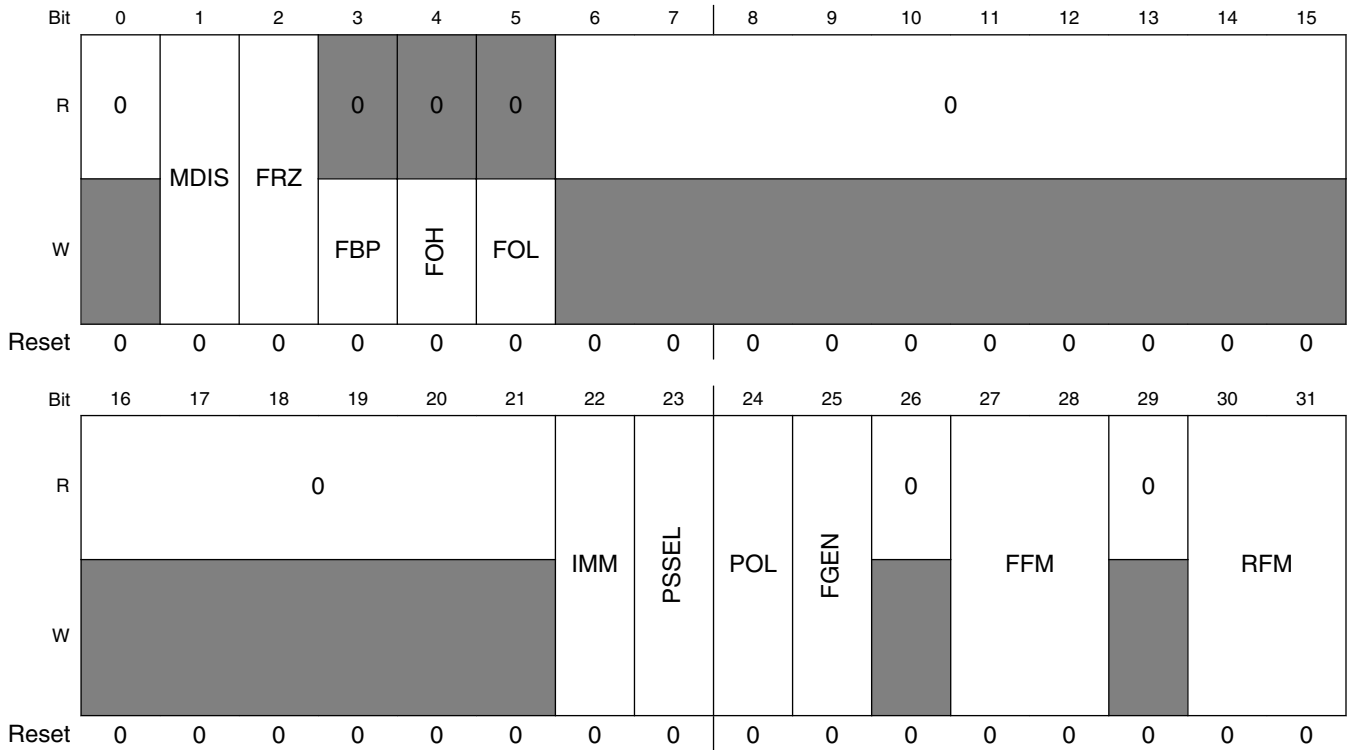
IGF memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
99C	Rising edge Threshold Register (IGF_RTHR38)	32	R/W	0000_0000h	39.5.4/1929
9C0	Module Configuration Register (IGF_MCR39)	32	R/W	0000_0000h	39.5.1/1924
9C4	Module Status Register (IGF_MSR39)	32	R/W	0000_0000h	39.5.2/1927
9DC	Rising edge Threshold Register (IGF_RTHR39)	32	R/W	0000_0000h	39.5.4/1929
A00	Module Configuration Register (IGF_MCR40)	32	R/W	0000_0000h	39.5.1/1924
A04	Module Status Register (IGF_MSR40)	32	R/W	0000_0000h	39.5.2/1927
A1C	Rising edge Threshold Register (IGF_RTHR40)	32	R/W	0000_0000h	39.5.4/1929
A40	Module Configuration Register (IGF_MCR41)	32	R/W	0000_0000h	39.5.1/1924
A44	Module Status Register (IGF_MSR41)	32	R/W	0000_0000h	39.5.2/1927
A5C	Rising edge Threshold Register (IGF_RTHR41)	32	R/W	0000_0000h	39.5.4/1929
A80	Module Configuration Register (IGF_MCR42)	32	R/W	0000_0000h	39.5.1/1924
A84	Module Status Register (IGF_MSR42)	32	R/W	0000_0000h	39.5.2/1927
A9C	Rising edge Threshold Register (IGF_RTHR42)	32	R/W	0000_0000h	39.5.4/1929
AC0	Module Configuration Register (IGF_MCR43)	32	R/W	0000_0000h	39.5.1/1924
AC4	Module Status Register (IGF_MSR43)	32	R/W	0000_0000h	39.5.2/1927
ADC	Rising edge Threshold Register (IGF_RTHR43)	32	R/W	0000_0000h	39.5.4/1929
B00	Module Configuration Register (IGF_MCR44)	32	R/W	0000_0000h	39.5.1/1924
B04	Module Status Register (IGF_MSR44)	32	R/W	0000_0000h	39.5.2/1927
B1C	Rising edge Threshold Register (IGF_RTHR44)	32	R/W	0000_0000h	39.5.4/1929
B40	Module Configuration Register (IGF_MCR45)	32	R/W	0000_0000h	39.5.1/1924
B44	Module Status Register (IGF_MSR45)	32	R/W	0000_0000h	39.5.2/1927
B5C	Rising edge Threshold Register (IGF_RTHR45)	32	R/W	0000_0000h	39.5.4/1929

39.5.1 Module Configuration Register (IGF_MCRn)

The Module Configuration register (MCR) provides access to low power, debug, and filter control bits.

Address: 0h base + 0h offset + (64d × i), where i=0d to 45d



IGF_MCRn field descriptions

Field	Description
0 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
1 MDIS	Module Disable The MDIS bit forces the filter to enter a low power state. After this bit is asserted the clock for the module is removed thus filter operation is not possible. The access to filter registers is allowed. 0 Input Glitch Filter is not in low power mode due to MDIS bit 1 Input Glitch Filter is in low power mode
2 FRZ	Freeze bit for debug operation The FRZ bit enables the Input Glitch Filter to enter freeze mode (debug mode) when a system level signal request the module enters freeze (debug) mode. Once in freeze mode the filter operation is disabled meaning that transitions on the input signal will not propagate through the filter output. The bypass FBP bit can be used by the software to force the filter output to be the same as the filter input. In this mode FOH and FOL also can be used to define the filter output state.

Table continues on the next page...

IGF_MCRn field descriptions (continued)

Field	Description
	0 Freeze mode disabled 1 Freeze mode enabled
3 FBP	Force Bypass This bit is self-negated thus is only read as zero. Writing 1 to this bit while in freeze mode forces the filter to bypass the value in the input to the filter output, thus allowing a software control on the filter. This bit is functional only in freeze mode. 0 no filter bypass 1 transfer input signal value to the filter output
4 FOH	Force filter output high If set, this bit forces the Input Glitch Filter output to '1'. This bit is functional only in debug mode. It is self-negated thus always read as zero. 0 no action 1 forces the filter output to '1'
5 FOL	Force filter output low If set this bit forces the Input Glitch Filter output to '0'. This bit is functional only in debug mode. It is self-negated thus always read as zero. 0 no action 1 forces the filter output to '0'
6–21 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
22 IMM	Immediate edge propagation control bit The IMM bit controls the propagation of an edge through the filter. If asserted, this bit defines that an edge at the filter input propagates through the filter output independent of the prescaler settings. In bypass mode the propagation delay is three system clock cycles. If negated, the signal propagation depends upon the prescaler settings. The filter hardware will only allow changes to this bit if the filter is disabled, FGEN = 0 0 edge propagation depends on prescaler 1 edge propagation within three system clock cycles
23 PSSEL	Prescaler selection bit The PSSEL selects if the prescaler used by the Input Glitch Filter is from internal source (internal prescaler counter) or from external source. The filter hardware will only allow changes to this bit if the filter is disabled, FGEN=0. 0 internal prescaler selected 1 external prescaler selected
24 POL	Output polarity bit This bit sets the polarity of the filter output to inverted or not inverted. The polarity control implemented by the POL bit is applied at the filter output stage thus also changing the reset state of the output, even if the filter is disabled by the FGEN bit. The filter hardware will only allow changes to this bit if the filter is disabled, FGEN=0.

Table continues on the next page...

IGF_MCRn field descriptions (continued)

Field	Description
	0 not inverted output 1 inverted output
25 FGEN	Filter global enable This control bit enables the filter operation, meaning that if the filter is enabled the filtering selected types are applied to the rising and falling edges of the input signal. If the filter is not enabled, then the filter output remains unchanged independent of the filter input signal. Note that the filter output state is affected by the POL bit when the filter is disabled. 0 filter is disabled: filter output keeps current state. 1 filter is enabled: filtering is applied to selected edges.
26 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
27–28 FFM	Falling edge filter type selection This bit field selects the filter type for the filter input signal falling edge. The filter hardware will only allow changes to this bit if the filter is disabled, FGEN=0. 00 bypass: edge is propagated to filter output without any filtering in three system clock cycles 01 windowing: windowing filter is selected for the falling edge 10 integrating: integrating filter is selected for the falling edge 11 integrating-hold: integrating-hold filter is selected for the falling edge
29 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
30–31 RFM	Rising edge filter type selection This bit field selects the filter type for the input signal rising edge. The filter hardware will only allow changes to this bit if the filter is disabled, FGEN=0. 00 bypass: edge is propagated to filter output without any filtering in three system clock cycles 01 windowing: windowing filter is selected for the rising edge 10 integrating: integrating filter is selected for the rising edge 11 integrating-hold: integrating-hold filter is selected for the rising edge

39.5.2 Module Status Register (IGF_MSRn)

The Module status register (MSR) provides access to filter status bits and corresponding clear bits as well as access to the state of filter input and output signals.

Address: 0h base + 4h offset + (64d × i), where i=0d to 45d

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0										WEDGE	FEDGE	RNDET	ENDET	FLI	FLO
W											w1c	w1c	w1c	w1c		
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

IGF_MSRn field descriptions

Field	Description
0–25 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
26 WEDGE	Filter is active waiting for an edge This bit is set when the filter is enabled but the internal counter is not active. It indicates that the filter is waiting for an edge to occur at the filter input. It is set by the Input Glitch Filter hardware and cleared by the software by writing 1 to the bit. Since the intention of this bit is to capture the dynamic of the filter hardware, it should be cleared after the read thus being able to capture a new event. 0 Input Glitch Filter is disabled or the internal counter is active 1 Input Glitch Filter is waiting for an edge
27 FEDGE	Filter is active processing an edge This bit is set during the filter active state thus when the filter internal counter is counting, processing an edge in a filtering mode. It is cleared by the software by writing 1 to the bit. Since the intention of this bit is to capture the dynamic of the filter hardware, it should be cleared after the read thus being able to capture a new event. 0 Input Glitch Filter internal counter is not active 1 Input Glitch Filter internal counter is active

Table continues on the next page...

IGF_MSR_n field descriptions (continued)

Field	Description
28 RNDET	Rise noise detected bit This bit indicates that the filter filtered out noise in the input signal after a rising edge was detected. The criteria to set this bit is when a rising edge is detected thus starting the filter counter and before this edge pass through the filter (if filter is in integrating filter types) or during a window (if filter is in windowing filter types) another edge occurs in the input signal. This bit is set by a noise detection and cleared by the software by writing 1 to this bit. 0 noise was not detected 1 noise was detected after a rising edge
29 FNDET	Fall noise detected bit This bit indicates that the filter filtered out noise in the input signal after a falling edge was detected. The criteria to set this bit is when a falling edge is detected, thus starting the filter counter, and, if the filter is in integrating types, before this edge passes through the filter, or, in windowing types, when another edge occurs in the input signal during a window. This bit is set by a noise detection and cleared by the software by writing 1 to this bit. 0 noise was not detected 1 noise was detected after a falling edge
30 FLI	Filter input The FLI bit follows the state of the filter input value as captured by the filter input logic, which is affected by the IMM bit and prescaler settings.
31 FLO	Filter output The FLO bit follows the state of the filter output value considering the POL bit state.

39.5.3 Prescaler Register (IGF_PRESR_n)

The prescaler register (PRESR) provides access to the prescaler bits.

Address: 0h base + 10h offset + (2048d × i), where i=0d to 1d

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0																						FPRE									
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	

IGF_PRESR_n field descriptions

Field	Description
0–22 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
23–31 FPRE	Filter prescaler The prescaler defines the rate of the filter counter FCOUNT. The filter counter clock rate is defined by the formula: Filter counter clock = module clock / (FPRE + 1).

39.5.4 Rising edge Threshold Register (IGF_RTHRn)

The Rising edge Threshold register (RTHR) provides access to the rising edge threshold bits.

Address: 0h base + 1Ch offset + (64d × i), where i=0d to 45d

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0								RTH																							
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	

IGF_RTHRn field descriptions

Field	Description
0–7 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
8–31 RTH	Rising edge threshold This bit field defines the filter counter threshold when a rising edge occurs is being filtered.

39.5.5 Falling edge Threshold Register (IGF_FTTHR)

The Falling edge Threshold register (FTTHR) provides access to the falling edge threshold bits.

Address: 0h base + 20h offset = 20h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0								FTH																							
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	

IGF_FTTHR field descriptions

Field	Description
0–7 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
8–31 FTH	Falling edge threshold This bit field defines the filter counter threshold when a falling edge is being filtered.

39.6 Functional description

The following sections describe the Input Glitch Filter module functionality.

39.6.1 Input double sampling

The filter input is double sampled for clock domain translation thus avoiding asynchronous signals to propagate through the filter internal logic. This stage is composed by two flip-flops connected back to back. The delay caused by this logic is therefore two clock cycles.

Note that due to the synchronous nature of the Input Glitch Filter design, if a glitch smaller than one clock cycle occurs at the filter input it will not propagate to the filter internal logic and thus will not propagate to the filter output. Also, if clock is not applied to the filter (i.e. in low power mode) any transition at the filter input will not propagate to the filter output.

39.6.2 Filter types

The Input Glitch Filter implements three main filter types:

- Windowing
- Integrating
- Integrating hold

These filter types can be selected for the rising and falling edges independently, thus allowing different filter types to be selected for different edges (i.e. rising edge filtered with windowing filter and falling edge filtered with integrating).

39.6.2.1 Windowing filter

This filter type detects an edge (positive or negative edge) at the filter input and then mask out detection of subsequent edges over a period of time, which is defined as the blanking period, thus effectively behaving as a high pass filter. After an edge is detected the level at the filter input is propagated to the filter output within three clock cycles. The delay is caused by the synchronization stage at the input and by the filter output register stage. During the blanking period any transition at the filter input is disregarded. After the blanking period has elapsed, the filter compares the input level state with the filter output state. In case they are different a transition is inferred and re-evaluate according to the filter type assigned to that transition (positive or negative edges). Since, in this example, the windowing filter type is programmed for both edges, the inferred negative edge is propagated to the filter output and another blanking period is initiated. The figure below describes the windowing filter type behavior.

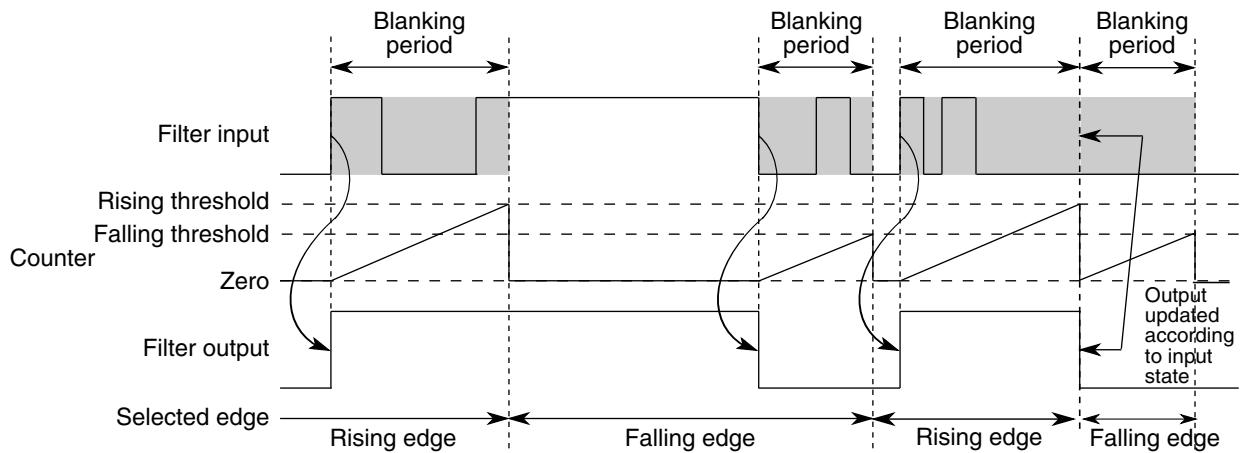


Figure 39-148. Windowing filter type

39.6.2.2 Integrating filter

The integrating filter behaves like a low-pass filter, thus removing high frequency bursts from the input signal. The input and output signals are compared and if they are at different states an internal filter counter is enabled.

The direction of the counter (up or down counting) is defined by the states of the input and output filter signals as follows:

- Filter output is at the same state as the filter input: down-counting
- Filter output is at different state as the filter input: up-counting

The counter is compared with the rising or falling threshold register according to:

- Rising threshold register is selected if the rising edge is being filtered.
- Falling threshold register is selected if the falling edge is being filtered.

The filter output changes state to the same state as the filter input:

- If filter counter reaches the selected threshold for the edge being filtered.

The internal counter is disabled and cleared:

- If filter counter reaches the selected threshold for the edge being filtered.

The figure below describes an example of the Integrating filter type operation.

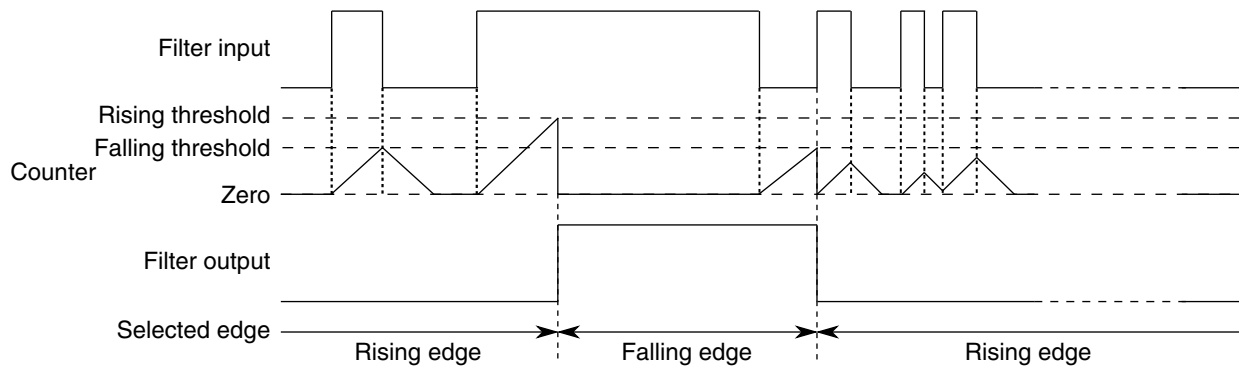


Figure 39-149. Integrating filter type

39.6.2.3 Integrating-hold filter

The integrating-hold filter type is similar to the integrating filter. The difference is the counter, which only counts in up direction instead of up-down in the integrating mode. In this filter configuration the input and output signals are compared and if they are at different states the filter counter is enabled. This counter is controlled by the state of the filter input and filter output signals as follows:

- Filter output is at the same state as the filter input: counter is in hold state (no-counting).
- Filter output is at different state as the filter input: counter is counting up.

The counter is compared with the rising or falling threshold register according to:

- Rising threshold register is selected if the rising edge is being filtered.
- Falling threshold register is selected if the falling edge is being filtered.

The filter output changes state to the same state as the filter input:

- If the filter counter reaches the selected threshold for the edge being filtered.

The internal counter is disabled and cleared:

- If the filter counter reaches the selected threshold for the edge being filtered.

The figure below shows an example of the Integrating-hold filter type operation.

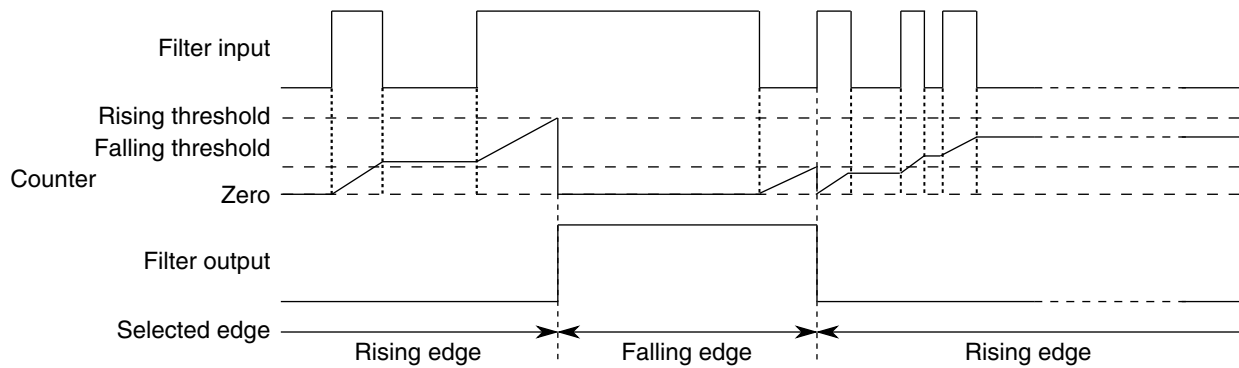


Figure 39-150. Integrating-hold filter type

39.6.2.4 Bypass filter

In the bypass filter type, the Input Glitch Filter immediately propagates any transition on the filter input to the filter output. This filter type does not provide any protection against glitches in the input signal. It is intended to be used in situations where the input signal is already filtered, or it is a glitch free signal, or the filter output is connected to a module that already has filtering capabilities. The bypass filter type is independently selected for rising or falling edges. Thus rising edges may be not filtered by the bypass filter type while the falling edges are filtered by any of the available filter types.

The bypass filter presents a delay in the edge propagation through the filter that is caused by the double sampling at the filter input and by the output filter register. Therefore, the total delay is three clock cycles.

39.6.2.5 Combined filter

The Input Glitch Filter allows the combination of filter types for different edges, positive and negative. As an example, the rising edges may be filtered using the windowing filter while the falling edges are filtered with an integrating filter. This configuration of the Input Glitch Filter is provided for situations where the noise present in the rising edges is different from the noise in the falling edges. Another use case is when the delay requirements of the input signal is different for rising and falling edges.

The figure below shows the combined filter operation. The windowing filter type is applied to the rising edges. The integrating filter is used for the falling edges.

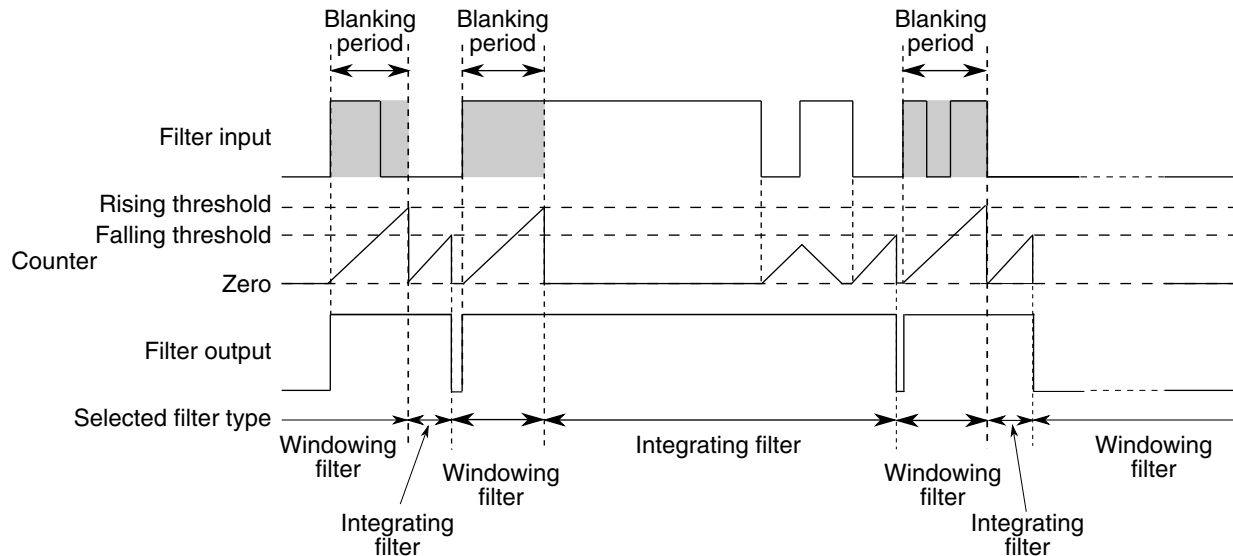


Figure 39-151. Combined filter type (windowing + integrating)

39.6.3 Application

This section describes several applications for the Input Glitch filter module.

39.6.3.1 Application use case 1: integrating filter

The integrating filter type is particularly useful in applications where glitches occur spuriously while the signal is expected to be in a steady state (either 1 or 0). Choosing correct up and down filter clock rates can ensure that the effects of spurious glitches can be completely eliminated, whilst minimizing propagation delays.

The application must be able to tolerate delays between the actual edge occurrence and the time that the signal is propagated through the filter logic. Because the delays are constants defined by the detection thresholds, application software may compensate for them, provided little or no noise occurs on the detected edge.

39.6.3.2 Application use case 2: windowing filter

The windowing filter type is particularly useful for removing noise bounce produced by a switch or sensor when its signal changes state. In this case, there is no propagation delay (except for the filter input synchronization logic and output register) between the filter input and filter output signals. This type of filter would not necessarily provide much immunity to spurious noise that occurred during the steady state portion of the signal. In

order to optimize the characteristics of this filter, the blanking period should be of the same magnitude as the expected input signal rate. Signals that change frequency can be accommodated by software by dynamically updating the blanking period.

39.6.3.3 Application use case 3: windowing filter

Safety applications may need to compare the inputs obtained from different input pins, but derived from a single external source. This form of redundancy may detect non-common mode faults that may occur in the portions of the SoC that route and process the incoming signal. Having two different instances of windowing filter used on each input pin will ensure that the signal edge timing for both input paths is equal. Applying different filter characteristics for each input path can enhance the ability to detect SoC fault conditions.

39.6.3.4 Application use case 4: combined filter

Some applications may have rising edge signals that exhibit different characteristics to falling edge signals which might require different filtering to optimize the filtering effect. For example the “first” rising edge may represent a true real time event with a fast slew rate that must be captured without delay, while the falling edge signal may have a slow slew rate with many noisy transitions. The Input Glitch Filter can apply, for example, a windowing filter on the rising edge to ensure no loss of real time information and an integrating filter on the falling edge to eliminate noise from being detected as a new rising transition.

39.6.3.5 Application use case 5: prescaler settings in windowing filter

The prescaler settings influence the timing for an edge to propagate through the filter. In windowing filter the IMM bit in the MCR register controls the way an edge is propagated through the filter. The figure below describes the windowing filter operation with regard to the IMM bit settings.

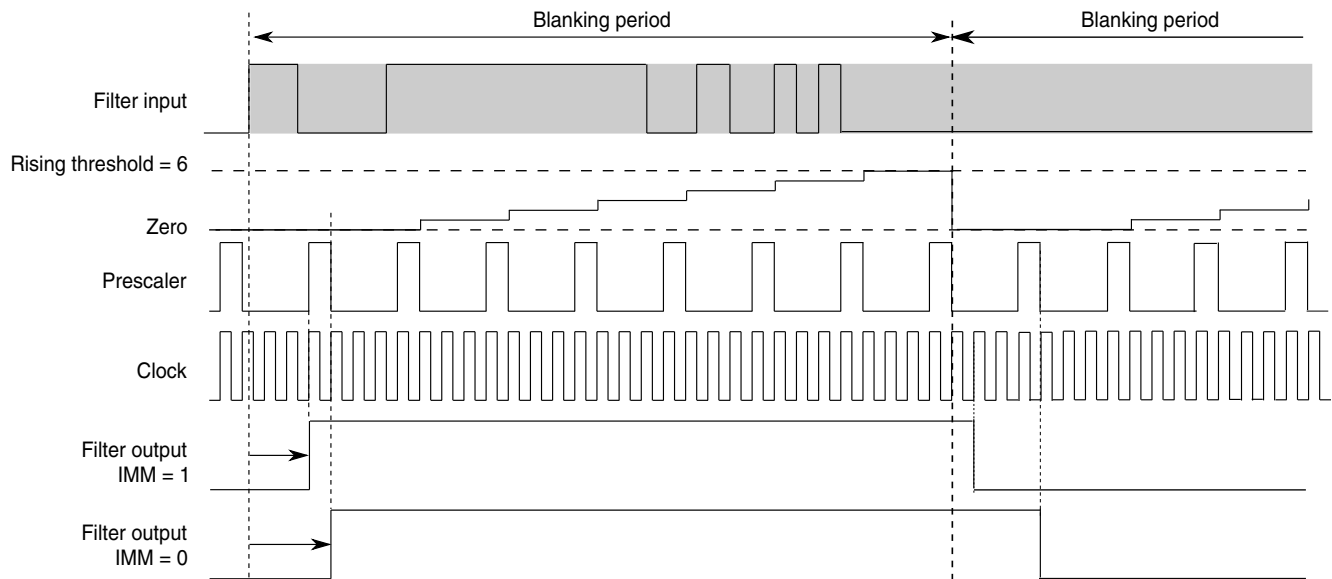


Figure 39-152. Input signal propagation through the filter in windowing filter type

If $IMM = 0$ then the input signal edges are propagated according to the prescaler ticks. If $IMM = 1$ then the input edges are propagated immediately (within 3 clock cycles) to the filter output.

39.6.3.6 Application use case 6: prescaler settings in bypass

The prescaler settings influence the timing for an input signal edge to propagate through the filter. In bypass, the IMM bit in the MCR register controls the way an edge is propagated through the filter. If $IMM = 0$, the prescaled clock is used to sample the input filter signal. The resolution the input signal is sampled is therefore related to the prescaler frequency. Glitches narrower than the prescaler period that are not aligned with the prescaler clock are not captured by the filter. If $IMM = 1$, the system clock is used to sample the filter input signal instead of the prescaled clock. In this case, the resolution of the sampled input signal is the system clock.

The figure below describes the filter input signal sampling behavior for $IMM = 0$ and $IMM = 1$.

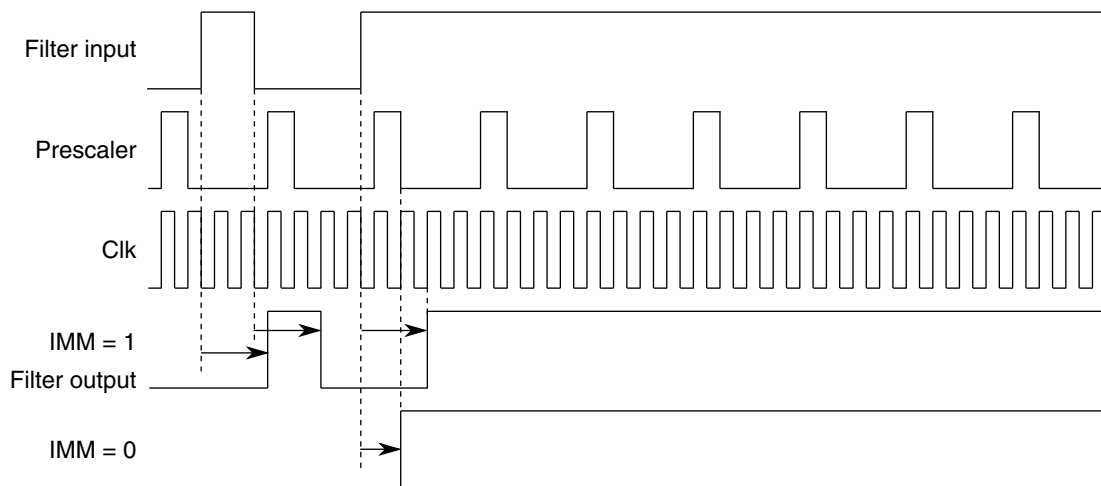


Figure 39-153. Input signal propagation through the filter in bypass

39.6.4 Programming guidelines

The following generalized procedure may be used to configure the Input Glitch Filter Block:

1. Program the SoC pin mux logic to route an external signal to the input of the selected glitch filter.
2. Enable the destination module to receive the Input Glitch filter output.
3. Enable the clock source at the SoC level.
4. Program the Glitch Filter counter prescaler(s) to the desired rate.
5. Select the filter type for each edge using MCR[FFM] and MCR[RFM].
6. Clear any status bits if necessary.
7. Enable the filter using MCR[FGEN]

39.6.5 On-the-fly filter reconfiguration

The architecture of the Input Glitch Filter allows for on the fly reconfiguration of filter parameters such as rise and fall threshold values.

If a parameter is changed it influences the filter operation immediately after the new parameter value was written. The main problem when changing parameters, such as a threshold value, when the filter is enabled is that the filter counter could lose a match due to the parameter change. A great or equal comparator is used in the threshold match logic which allows correct filter operation when the threshold parameters are changed. The figure below shows an example of changing the rising edge parameter value. Note that in this case the match occurs immediately after the parameter is changed since the new parameter value is below the current counter value.

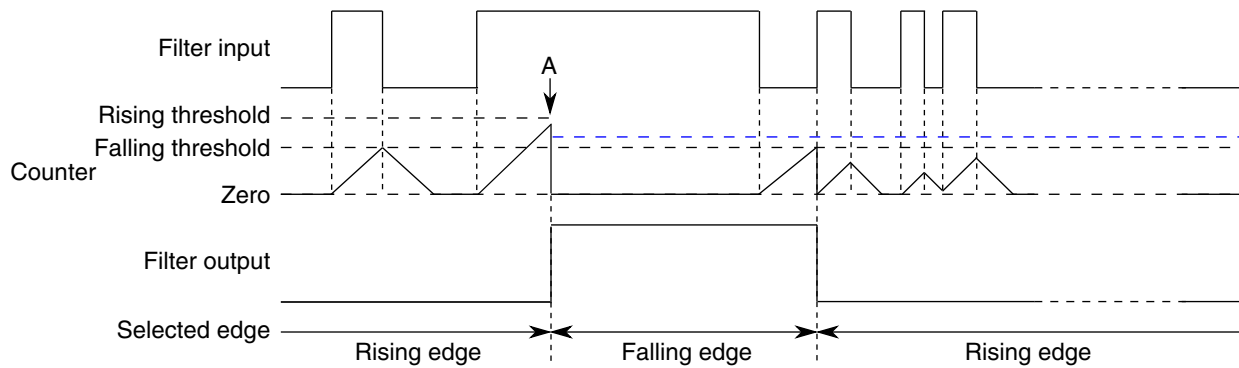


Figure 39-154. On-the-fly rising threshold parameter change

In the case shown in the figure above the match occurs immediately after the new rising threshold is programmed at 'A'.

NOTE

Note that when changing the threshold values the match may not occur when the counter is equal to the threshold value (the old or the new programmed value). Instead, the match may occur in between the old and new programmed thresholds when the threshold re-programming instant A is done when the counter is greater than the new value. This is a transitory effect which may occur when the threshold value is changed and should be considered by the application.

39.7 Power saving mode

The Input Glitch Filter implements power saving mode thus allowing to reduce power consumption when device inputs connected to the filter are not used or the filter is not necessary to be functioning in a specific SoC mode of operation.

39.7.1 STOP mode

There are two ways for the Input Glitch filter to enter low power mode:

- MDIS bit
- STOP mode system request

The MDIS control bit in the MCR register, when set, disables the filter logic by stopping the module clock signals and thus halting the internal filter counter which reduces the dynamic power consumption of the module. The MDIS bit provides a selective control of the low power operation of each one of the Input Glitch Filter instances in the device.

The STOP mode system request is similar to MDIS operation with exception that it applies to all Input Glitch Filter instances in the device. It does not need to be enabled by any control bit in the Input Glitch filter module. The low power mode in this case is controlled by the assertion of the STOP signal which is a system wide low power control signal.

Chapter 40

FlexCAN (CAN)

40.1 Chip-specific FlexCAN information

This chip has four FlexCAN modules.

Each FlexCAN instance supports 64 message buffers.

40.1.1 FlexCAN RAM

On this chip, each FlexCAN module instance has four 1.5 KB dedicated RAM blocks.

40.1.2 FlexCAN reset

The FlexCAN architecture has two sources of reset: one resets the entire FlexCAN system and the other resets only the Host Control Interface (CHI). This chip supports only the global reset, which is driven by the internal reset of the device.

40.1.3 CAN_MCR[SUPV] and access to FlexCAN registers

On this chip, regardless of the value of CAN_MCR[SUPV], all core accesses to FlexCAN registers occur in supervisor mode.

40.2 Introduction

The FlexCAN module is a communication controller implementing the CAN protocol according to the CAN 2.0 B protocol specification. A general block diagram is shown in the following figure, which describes the main subblocks implemented in the FlexCAN module, including one associated memory for storing message buffers, Receive (Rx)

Global Mask registers, Receive Individual Mask registers, Receive FIFO filters, and Receive FIFO ID filters. The functions of the submodules are described in subsequent sections.

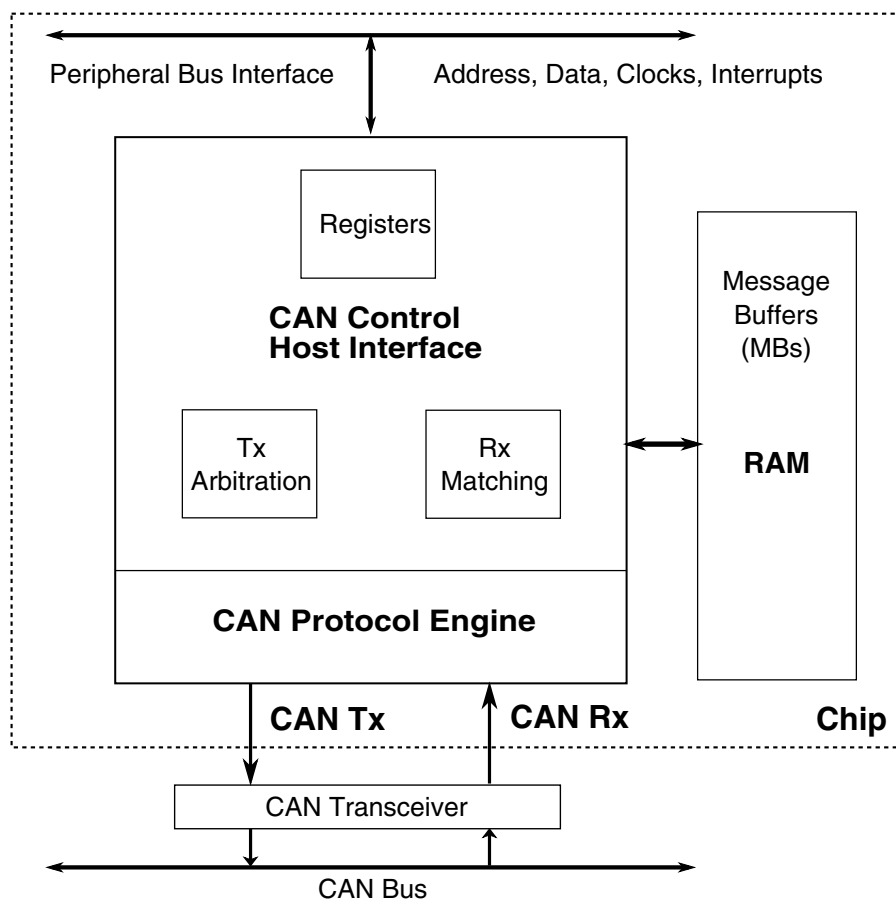


Figure 40-1. FlexCAN block diagram

40.2.1 Overview

The CAN protocol was primarily designed to be used as a vehicle serial data bus, meeting the specific requirements of this field:

- Real-time processing
- Reliable operation in the EMI environment of a vehicle
- Cost-effectiveness
- Required bandwidth

The FlexCAN module is a full implementation of the CAN protocol specification, Version 2.0 B, which supports both standard and extended message frames. The message buffers are stored in an embedded RAM dedicated to the FlexCAN module. See the chip-specific FlexCAN information for the actual number of message buffers configured in the chip.

The CAN Protocol Engine (PE) submodule manages the serial communication on the CAN bus:

- Requesting RAM access for receiving and transmitting message frames
- Validating received messages
- Performing error handling

The Controller Host Interface (CHI) sub-module handles message buffer selection for reception and transmission, taking care of arbitration and ID matching algorithms.

The Bus Interface Unit (BIU) sub-module controls the access to and from the internal interface bus, in order to establish connection to the CPU and to other blocks. Clocks, address and data buses, interrupt outputs and test signals are accessed through the BIU.

40.2.2 FlexCAN module features

The FlexCAN module includes these distinctive legacy features:

- Full implementation of the CAN protocol specification, Version 2.0 B
 - Standard data and remote frames
 - Extended data and remote frames
 - Zero to eight bytes data length
 - Programmable bit rate up to 1 Mb/sec
 - Content-related addressing
- Compliant with the ISO 11898-1 standard
- Flexible mailboxes of zero to eight bytes data length
- Each mailbox configurable as receive or transmit, all supporting standard and extended messages
- Individual Rx Mask registers per mailbox
- Full-featured Rx FIFO with storage capacity for up to six frames and automatic internal pointer handling
- Transmission abort capability
- Programmable clock source to the CAN Protocol Interface, either bus clock or crystal oscillator
- Unused structures space can be used as general purpose RAM space

- Listen-Only mode capability
- Programmable Loop-Back mode supporting self-test operation
- Programmable transmission priority scheme: lowest ID, lowest buffer number, or highest priority
- Time stamp based on 16-bit free-running timer
- Global network time, synchronized by a specific message
- Maskable interrupts
- Independence from the transmission medium (an external transceiver is assumed)
- Short latency time due to an arbitration scheme for high-priority messages
- Low power modes

New major features are also provided:

- Remote request frames may be handled automatically or by software
- CAN bit time settings and configuration bits can only be written in Freeze mode
- Tx mailbox status (Lowest priority buffer or empty buffer)
- Identifier Acceptance Filter Hit Indicator (IDHIT) register for received frames
- SYNCH bit available in Error in Status 1 register to inform that the module is synchronous with CAN bus
- CRC status for transmitted message
- Rx FIFO Global Mask register
- Selectable priority between mailboxes and Rx FIFO during matching process
- Powerful Rx FIFO ID filtering, capable of matching incoming IDs against either 128 extended, 256 standard, or 512 partial (8 bit) IDs, with up to 32 individual masking capability
- 100% backward compatibility with previous FlexCAN version
- Supports detection and correction of errors in memory read accesses. Each byte of FlexCAN memory is associated to 5 parity bits and the error correction mechanism assures that in this 13-bit word, errors in one bit can be corrected (corrected errors) and errors in 2 bits can be detected but not corrected (non-corrected errors).

40.2.3 Modes of operation

The FlexCAN module has these functional modes:

- Normal mode (User or Supervisor):

In Normal mode, the module operates receiving and/or transmitting message frames, errors are handled normally, and all CAN Protocol functions are enabled. User and Supervisor Modes differ in the access to some restricted control registers.

- Freeze mode:

Freeze mode is enabled when the FRZ bit in MCR is asserted. If enabled, Freeze mode is entered when MCR[HALT] is set or when Debug mode is requested at chip level and MCR[FRZ_ACK] is asserted by the FlexCAN. In this mode, no transmission or reception of frames is done and synchronicity to the CAN bus is lost. See [Freeze mode](#) for more information.

- Listen-Only mode:

The module enters this mode when the LOM field in the Control 1 Register is asserted. In this mode, transmission is disabled, all error counters are frozen, and the module operates in a CAN Error Passive mode. Only messages acknowledged by another CAN station will be received. If FlexCAN detects a message that has not been acknowledged, it will flag a BIT0 error (without changing the REC), as if it was trying to acknowledge the message.

- Loop-Back mode:

The module enters this mode when the LPB field in the Control 1 Register is asserted. In this mode, FlexCAN performs an internal loop back that can be used for self-test operation. The bit stream output of the transmitter is internally fed back to the receiver input. The Rx CAN input pin is ignored and the Tx CAN output goes to the recessive state (logic '1'). FlexCAN behaves as it normally does when transmitting and treats its own transmitted message as a message received from a remote node. In this mode, FlexCAN ignores the bit sent during the ACK slot in the CAN frame acknowledge field to ensure proper reception of its own message. Both transmit and receive interrupts are generated.

For low-power operation, the FlexCAN module has:

- Module Disable mode:

This low-power mode is entered when the MDIS bit in the MCR Register is asserted by the CPU and the LPM_ACK is asserted by the FlexCAN. When disabled, the module requests to disable the clocks to the CAN Protocol Engine and Controller Host Interface submodules. Exit from this mode is done by negating the MDIS bit in the MCR register. See [Module Disable mode](#) for more information.

- Doze mode:

This low power mode is entered when the DOZE bit in MCR is asserted and Doze mode is requested at chip level and the LPM_ACK bit in the MCR Register is asserted by the FlexCAN. When in Doze mode, the module requests to disable the clocks to the CAN Protocol Engine and the CAN Controller-Host Interface submodules. Exit from this mode happens when the DOZE bit in MCR is negated, when the chip is removed from Doze mode. See [Doze mode](#) for more information.

- Stop mode:

This low power mode is entered when Stop mode is requested at chip level and the LPM_ACK bit in the MCR Register is asserted by the FlexCAN. When in Stop Mode, the module puts itself in an inactive state and then informs the CPU that the clocks can be shut down globally. Exit from this mode happens when the Stop mode request is removed. See [Stop mode](#) for more information.

40.3 FlexCAN signal descriptions

The FlexCAN module has two I/O signals connected to the external chip pins. These signals are summarized in the following table and described in more detail in the next subsections.

Table 40-1. FlexCAN signal descriptions

Signal	Description	I/O
CAN Rx	CAN Receive Pin	Input
CAN Tx	CAN Transmit Pin	Output

40.3.1 CAN Rx

This pin is the receive pin from the CAN bus transceiver. Dominant state is represented by logic level 0. Recessive state is represented by logic level 1.

40.3.2 CAN Tx

This pin is the transmit pin to the CAN bus transceiver. Dominant state is represented by logic level 0. Recessive state is represented by logic level 1.

40.4 Memory map/register definition

This section describes the registers and data structures in the FlexCAN module. The base address of the module depends on the particular memory map of the chip.

40.4.1 FlexCAN memory mapping

The complete memory map for a FlexCAN module is shown in the following table.

The address space occupied by FlexCAN has 128 bytes for registers starting at the module base address, followed by embedded RAM starting at address 0x0080.

Each individual register is identified by its complete name and the corresponding mnemonic. The access type can be Supervisor (S) or Unrestricted (U). Most of the registers can be configured to have either Supervisor or Unrestricted access by programming the SUPV field in the MCR register. These registers are identified as S/U in the Access column of [Table 40-2](#).

Table 40-2. Register access and reset information

Register	Access type	Affected by hard reset	Affected by soft reset
Module Configuration Register (MCR)	S	Yes	Yes
Control 1 register (CTRL1)	S/U	Yes	No
Free Running Timer register (TIMER)	S/U	Yes	Yes
Rx Mailboxes Global Mask register (RXMGMASK)	S/U	No	No
Rx Buffer 14 Mask register (RX14MASK)	S/U	No	No
Rx Buffer 15 Mask register (RX15MASK)	S/U	No	No
Error Counter Register (ECR)	S/U	Yes	Yes
Error and Status 1 Register (ESR1)	S/U	Yes	Yes
Interrupt Masks 2 register (IMASK2)	S/U	Yes	Yes
Interrupt Masks 1 register (IMASK1)	S/U	Yes	Yes
Interrupt Flags 2 register (IFLAG2)	S/U	Yes	Yes
Interrupt Flags 1 register (IFLAG1)	S/U	Yes	Yes
Control 2 Register (CTRL2)	S/U	Yes	No

Table continues on the next page...

Table 40-2. Register access and reset information (continued)

Register	Access type	Affected by hard reset	Affected by soft reset
Error and Status 2 Register (ESR2)	S/U	Yes	Yes
CRC Register (CRCR)	S/U	Yes	Yes
Rx FIFO Global Mask register (RXFGMASK)	S/U	No	No
Rx FIFO Information Register (RXFIR)	S/U	No	No
Message buffers	S/U	No	No
Rx Individual Mask Registers	S/U	No	No
Memory Error Control Register (MECR)	S/U	Yes	Yes
Error Injection Address Register (ERRIAR)	S/U	Yes	Yes
Error Injection Data Pattern Register (ERRIDPR)	S/U	Yes	Yes
Error Injection Parity Pattern Register (ERRIPPR)	S/U	Yes	Yes
Error Report Address Register (RERRAR)	S/U	Yes	Yes
Error Report Data Register (RERRDR)	S/U	Yes	Yes
Error Report Syndrome Register (RERRSYNR)	S/U	Yes	Yes
Error Status Register (ERRSR)	S/U	Yes	Yes

The FlexCAN module can store CAN messages for transmission and reception using mailboxes and Rx FIFO structures.

This module's memory map includes sixty-four 128-bit message buffers (MBs) that occupy the range from offset 0x80 to 0x47F .

CAN memory map

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
0	Module Configuration Register (CAN_MCR)	32	R/W	5980_000Fh	40.4.2/1953
4	Control 1 register (CAN_CTRL1)	32	R/W	0000_0000h	40.4.3/1958
8	Free Running Timer (CAN_TIMER)	32	R/W	0000_0000h	40.4.4/1961
10	Rx Mailboxes Global Mask Register (CAN_RXMGMASK)	32	R/W	FFFF_FFFFh	40.4.5/1962
14	Rx 14 Mask register (CAN_RX14MASK)	32	R/W	FFFF_FFFFh	40.4.6/1963
18	Rx 15 Mask register (CAN_RX15MASK)	32	R/W	FFFF_FFFFh	40.4.7/1964
1C	Error Counter (CAN_ECR)	32	R/W	0000_0000h	40.4.8/1964
20	Error and Status 1 register (CAN_ESR1)	32	R/W	0000_0000h	40.4.9/1966
24	Interrupt Masks 2 register (CAN_IMASK2)	32	R/W	0000_0000h	40.4.10/1970
28	Interrupt Masks 1 register (CAN_IMASK1)	32	R/W	0000_0000h	40.4.11/1970
2C	Interrupt Flags 2 register (CAN_IFLAG2)	32	R/W	0000_0000h	40.4.12/1971

Table continues on the next page...

CAN memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
30	Interrupt Flags 1 register (CAN_IFLAG1)	32	R/W	0000_0000h	40.4.13/1972
34	Control 2 register (CAN_CTRL2)	32	R/W	0080_0000h	40.4.14/1974
38	Error and Status 2 register (CAN_ESR2)	32	R/W	0000_0000h	40.4.15/1978
44	CRC Register (CAN_CRCR)	32	R	0000_0000h	40.4.16/1979
48	Rx FIFO Global Mask register (CAN_RXFGMASK)	32	R/W	FFFF_FFFFh	40.4.17/1980
4C	Rx FIFO Information Register (CAN_RXFIR)	32	R	Undefined	40.4.18/1981
880	Rx Individual Mask Registers (CAN_RXIMR0)	32	R/W	Undefined	40.4.19/1981
884	Rx Individual Mask Registers (CAN_RXIMR1)	32	R/W	Undefined	40.4.19/1981
888	Rx Individual Mask Registers (CAN_RXIMR2)	32	R/W	Undefined	40.4.19/1981
88C	Rx Individual Mask Registers (CAN_RXIMR3)	32	R/W	Undefined	40.4.19/1981
890	Rx Individual Mask Registers (CAN_RXIMR4)	32	R/W	Undefined	40.4.19/1981
894	Rx Individual Mask Registers (CAN_RXIMR5)	32	R/W	Undefined	40.4.19/1981
898	Rx Individual Mask Registers (CAN_RXIMR6)	32	R/W	Undefined	40.4.19/1981
89C	Rx Individual Mask Registers (CAN_RXIMR7)	32	R/W	Undefined	40.4.19/1981
8A0	Rx Individual Mask Registers (CAN_RXIMR8)	32	R/W	Undefined	40.4.19/1981
8A4	Rx Individual Mask Registers (CAN_RXIMR9)	32	R/W	Undefined	40.4.19/1981
8A8	Rx Individual Mask Registers (CAN_RXIMR10)	32	R/W	Undefined	40.4.19/1981
8AC	Rx Individual Mask Registers (CAN_RXIMR11)	32	R/W	Undefined	40.4.19/1981
8B0	Rx Individual Mask Registers (CAN_RXIMR12)	32	R/W	Undefined	40.4.19/1981
8B4	Rx Individual Mask Registers (CAN_RXIMR13)	32	R/W	Undefined	40.4.19/1981
8B8	Rx Individual Mask Registers (CAN_RXIMR14)	32	R/W	Undefined	40.4.19/1981
8BC	Rx Individual Mask Registers (CAN_RXIMR15)	32	R/W	Undefined	40.4.19/1981

Table continues on the next page...

CAN memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
8C0	Rx Individual Mask Registers (CAN_RXIMR16)	32	R/W	Undefined	40.4.19/1981
8C4	Rx Individual Mask Registers (CAN_RXIMR17)	32	R/W	Undefined	40.4.19/1981
8C8	Rx Individual Mask Registers (CAN_RXIMR18)	32	R/W	Undefined	40.4.19/1981
8CC	Rx Individual Mask Registers (CAN_RXIMR19)	32	R/W	Undefined	40.4.19/1981
8D0	Rx Individual Mask Registers (CAN_RXIMR20)	32	R/W	Undefined	40.4.19/1981
8D4	Rx Individual Mask Registers (CAN_RXIMR21)	32	R/W	Undefined	40.4.19/1981
8D8	Rx Individual Mask Registers (CAN_RXIMR22)	32	R/W	Undefined	40.4.19/1981
8DC	Rx Individual Mask Registers (CAN_RXIMR23)	32	R/W	Undefined	40.4.19/1981
8E0	Rx Individual Mask Registers (CAN_RXIMR24)	32	R/W	Undefined	40.4.19/1981
8E4	Rx Individual Mask Registers (CAN_RXIMR25)	32	R/W	Undefined	40.4.19/1981
8E8	Rx Individual Mask Registers (CAN_RXIMR26)	32	R/W	Undefined	40.4.19/1981
8EC	Rx Individual Mask Registers (CAN_RXIMR27)	32	R/W	Undefined	40.4.19/1981
8F0	Rx Individual Mask Registers (CAN_RXIMR28)	32	R/W	Undefined	40.4.19/1981
8F4	Rx Individual Mask Registers (CAN_RXIMR29)	32	R/W	Undefined	40.4.19/1981
8F8	Rx Individual Mask Registers (CAN_RXIMR30)	32	R/W	Undefined	40.4.19/1981
8FC	Rx Individual Mask Registers (CAN_RXIMR31)	32	R/W	Undefined	40.4.19/1981
900	Rx Individual Mask Registers (CAN_RXIMR32)	32	R/W	Undefined	40.4.19/1981
904	Rx Individual Mask Registers (CAN_RXIMR33)	32	R/W	Undefined	40.4.19/1981
908	Rx Individual Mask Registers (CAN_RXIMR34)	32	R/W	Undefined	40.4.19/1981
90C	Rx Individual Mask Registers (CAN_RXIMR35)	32	R/W	Undefined	40.4.19/1981
910	Rx Individual Mask Registers (CAN_RXIMR36)	32	R/W	Undefined	40.4.19/1981
914	Rx Individual Mask Registers (CAN_RXIMR37)	32	R/W	Undefined	40.4.19/1981

Table continues on the next page...

CAN memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
918	Rx Individual Mask Registers (CAN_RXIMR38)	32	R/W	Undefined	40.4.19/1981
91C	Rx Individual Mask Registers (CAN_RXIMR39)	32	R/W	Undefined	40.4.19/1981
920	Rx Individual Mask Registers (CAN_RXIMR40)	32	R/W	Undefined	40.4.19/1981
924	Rx Individual Mask Registers (CAN_RXIMR41)	32	R/W	Undefined	40.4.19/1981
928	Rx Individual Mask Registers (CAN_RXIMR42)	32	R/W	Undefined	40.4.19/1981
92C	Rx Individual Mask Registers (CAN_RXIMR43)	32	R/W	Undefined	40.4.19/1981
930	Rx Individual Mask Registers (CAN_RXIMR44)	32	R/W	Undefined	40.4.19/1981
934	Rx Individual Mask Registers (CAN_RXIMR45)	32	R/W	Undefined	40.4.19/1981
938	Rx Individual Mask Registers (CAN_RXIMR46)	32	R/W	Undefined	40.4.19/1981
93C	Rx Individual Mask Registers (CAN_RXIMR47)	32	R/W	Undefined	40.4.19/1981
940	Rx Individual Mask Registers (CAN_RXIMR48)	32	R/W	Undefined	40.4.19/1981
944	Rx Individual Mask Registers (CAN_RXIMR49)	32	R/W	Undefined	40.4.19/1981
948	Rx Individual Mask Registers (CAN_RXIMR50)	32	R/W	Undefined	40.4.19/1981
94C	Rx Individual Mask Registers (CAN_RXIMR51)	32	R/W	Undefined	40.4.19/1981
950	Rx Individual Mask Registers (CAN_RXIMR52)	32	R/W	Undefined	40.4.19/1981
954	Rx Individual Mask Registers (CAN_RXIMR53)	32	R/W	Undefined	40.4.19/1981
958	Rx Individual Mask Registers (CAN_RXIMR54)	32	R/W	Undefined	40.4.19/1981
95C	Rx Individual Mask Registers (CAN_RXIMR55)	32	R/W	Undefined	40.4.19/1981
960	Rx Individual Mask Registers (CAN_RXIMR56)	32	R/W	Undefined	40.4.19/1981
964	Rx Individual Mask Registers (CAN_RXIMR57)	32	R/W	Undefined	40.4.19/1981
968	Rx Individual Mask Registers (CAN_RXIMR58)	32	R/W	Undefined	40.4.19/1981
96C	Rx Individual Mask Registers (CAN_RXIMR59)	32	R/W	Undefined	40.4.19/1981

Table continues on the next page...

CAN memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
970	Rx Individual Mask Registers (CAN_RXIMR60)	32	R/W	Undefined	40.4.19/1981
974	Rx Individual Mask Registers (CAN_RXIMR61)	32	R/W	Undefined	40.4.19/1981
978	Rx Individual Mask Registers (CAN_RXIMR62)	32	R/W	Undefined	40.4.19/1981
97C	Rx Individual Mask Registers (CAN_RXIMR63)	32	R/W	Undefined	40.4.19/1981
AE0	Memory Error Control Register (CAN_MECCR)	32	R/W	800C_0080h	40.4.20/1982
AE4	Error Injection Address Register (CAN_ERRIAR)	32	R/W	0000_0000h	40.4.21/1984
AE8	Error Injection Data Pattern Register (CAN_ERRIDPR)	32	R/W	0000_0000h	40.4.22/1986
AEC	Error Injection Parity Pattern Register (CAN_ERRIPPR)	32	R/W	0000_0000h	40.4.23/1986
AF0	Error Report Address Register (CAN_RERRAR)	32	R	0000_0000h	40.4.24/1987
AF4	Error Report Data Register (CAN_RERRDR)	32	R	0000_0000h	40.4.25/1988
AF8	Error Report Syndrome Register (CAN_RERRSYNR)	32	R	0000_0000h	40.4.26/1989
AFC	Error Status Register (CAN_ERRSR)	32	R/W	0000_0000h	40.4.27/1991

40.4.2 Module Configuration Register (CAN_MCR)

This register defines global system configurations, such as the module operation modes and the maximum message buffer configuration.

Address: 0h base + 0h offset = 0h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R					NOTRDY			FRZACK				LPMACK				
W																
Reset	0	1	0	1	1	0	0	1	1	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0			0				0							
W			LPRIOEN	AEN			IDAM						MAXMB			
Reset	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1

CAN_MCR field descriptions

Field	Description
0 MDIS	Module Disable This bit controls whether FlexCAN is enabled or not. When disabled, FlexCAN disables the clocks to the CAN Protocol Engine and Controller Host Interface sub-modules. This is the only bit within this register not affected by soft reset. 0 Enable the FlexCAN module. 1 Disable the FlexCAN module.
1 FRZ	Freeze Enable The FRZ bit specifies the FlexCAN behavior when the HALT bit in the MCR Register is set or when Debug mode is requested at chip level . When FRZ is asserted, FlexCAN is enabled to enter Freeze mode. Negation of this bit field causes FlexCAN to exit from Freeze mode.

Table continues on the next page...

CAN_MCR field descriptions (continued)

Field	Description
	0 Not enabled to enter Freeze mode. 1 Enabled to enter Freeze mode.
2 RFEN	Rx FIFO Enable This bit controls whether the Rx FIFO feature is enabled or not. When RFEN is set, MBs 0 to 5 cannot be used for normal reception and transmission because the corresponding memory region (0x80-0xDC) is used by the FIFO engine as well as additional MBs (up to 32, depending on CTRL2[RFFN] setting) which are used as Rx FIFO ID Filter Table elements. RFEN also impacts the definition of the minimum number of peripheral clocks per CAN bit as described in the table "Minimum Ratio Between Peripheral Clock Frequency and CAN Bit Rate" (in section "Arbitration and Matching Timing"). This bit can be written only in Freeze mode because it is blocked by hardware in other modes. 0 Rx FIFO not enabled. 1 Rx FIFO enabled.
3 HALT	Halt FlexCAN Assertion of this bit puts the FlexCAN module into Freeze mode. The CPU should clear it after initializing the Message Buffers and Control Register. No reception or transmission is performed by FlexCAN before this bit is cleared. Freeze mode cannot be entered while FlexCAN is in a low power mode. 0 No Freeze mode request. 1 Enters Freeze mode if the FRZ bit is asserted.
4 NOTRDY	FlexCAN Not Ready This read-only bit indicates that FlexCAN is either in Disable mode , Doze mode, Stop mode or Freeze mode. It is negated once FlexCAN has exited these modes. 0 FlexCAN module is either in Normal mode, Listen-Only mode or Loop-Back mode. 1 FlexCAN module is either in Disable mode , Doze mode, Stop mode or Freeze mode.
5 Reserved	This field is reserved. Always write 0 to this field.
6 SOFTTRST	Soft Reset When this bit is asserted, FlexCAN resets its internal state machines and some of the memory mapped registers. The following registers are reset: MCR (except the MDIS bit), TIMER , ECR, ESR1, ESR2, IMASK1, IMASK2, IFLAG1, IFLAG2 and CRCR. Configuration registers that control the interface to the CAN bus are not affected by soft reset. The following registers are unaffected: CTRL1, CTRL2, all RXIMR registers, RXMGMASK, RX14MASK, RX15MASK, RXFGMASK, RXFIR, all Message Buffers . The SOFTTRST bit can be asserted directly by the CPU when it writes to the MCR Register. Because soft reset is synchronous and has to follow a request/acknowledge procedure across clock domains, it may take some time to fully propagate its effect. The SOFTTRST bit remains asserted while reset is pending, and is automatically negated when reset completes. Therefore, software can poll this bit to know when the soft reset has completed. Soft reset cannot be applied while clocks are shut down in a low power mode. The module should be first removed from low power mode, and then soft reset can be applied. 0 No reset request. 1 Resets the registers affected by soft reset.
7 FRZACK	Freeze Mode Acknowledge This read-only bit indicates that FlexCAN is in Freeze mode and its prescaler is stopped. The Freeze mode request cannot be granted until current transmission or reception processes have finished.

Table continues on the next page...

CAN_MCR field descriptions (continued)

Field	Description
	Therefore the software can poll the FRZACK bit to know when FlexCAN has actually entered Freeze mode. If Freeze Mode request is negated, then this bit is negated after the FlexCAN prescaler is running again. If Freeze mode is requested while FlexCAN is in a low power mode, then the FRZACK bit will be set only when the low-power mode is exited. See Section "Freeze Mode". NOTE: FRZACK will be asserted within 178 CAN bits from the freeze mode request by the CPU, and negated within 2 CAN bits after the freeze mode request removal (see Section "Protocol Timing"). 0 FlexCAN not in Freeze mode, prescaler running. 1 FlexCAN in Freeze mode, prescaler stopped.
8 SUPV	Supervisor Mode This bit configures the FlexCAN to be either in Supervisor or User mode. The registers affected by this bit are marked as S/U in the Access Type column of the module memory map. Reset value of this bit is 1, so the affected registers start with Supervisor access allowance only. This bit can be written only in Freeze mode because it is blocked by hardware in other modes. 0 FlexCAN is in User mode. Affected registers allow both Supervisor and Unrestricted accesses. 1 FlexCAN is in Supervisor mode. Affected registers allow only Supervisor access. Unrestricted access behaves as though the access was done to an unimplemented register location.
9 Reserved	This field is reserved. Always write 0 to this field.
10 WRNEN	Warning Interrupt Enable When asserted, this bit enables the generation of the TWRNINT and RWRNINT flags in the Error and Status Register. If WRNEN is negated, the TWRNINT and RWRNINT flags will always be zero, independent of the values of the error counters, and no warning interrupt will ever be generated. This bit can be written only in Freeze mode because it is blocked by hardware in other modes. 0 TWRNINT and RWRNINT bits are zero, independent of the values in the error counters. 1 TWRNINT and RWRNINT bits are set when the respective error counter transitions from less than 96 to greater than or equal to 96.
11 LPMACK	Low-Power Mode Acknowledge This read-only bit indicates that FlexCAN is in a low-power mode (Disable mode, Doze mode, Stop mode). A low-power mode cannot be entered until all current transmission or reception processes have finished, so the CPU can poll the LPMACK bit to know when FlexCAN has actually entered low power mode. NOTE: LPMACK will be asserted within 180 CAN bits from the low-power mode request by the CPU, and negated within 2 CAN bits after the low-power mode request removal (see Section "Protocol Timing"). 0 FlexCAN is not in a low-power mode. 1 FlexCAN is in a low-power mode.
12 Reserved	This field is reserved. Always write 0 to this field.
13 DOZE	Doze Mode Enable This bit defines whether FlexCAN is allowed to enter low-power mode when Doze mode is requested at chip level.

Table continues on the next page...

CAN_MCR field descriptions (continued)

Field	Description
	0 FlexCAN is not enabled to enter low-power mode when Doze mode is requested. 1 FlexCAN is enabled to enter low-power mode when Doze mode is requested.
14 SRXDIS	Self Reception Disable This bit defines whether FlexCAN is allowed to receive frames transmitted by itself. If this bit is asserted, frames transmitted by the module will not be stored in any MB, regardless if the MB is programmed with an ID that matches the transmitted frame, and no interrupt flag or interrupt signal will be generated due to the frame reception. This bit can be written only in Freeze mode because it is blocked by hardware in other modes. 0 Self reception enabled. 1 Self reception disabled.
15 IRMQ	Individual Rx Masking And Queue Enable This bit indicates whether Rx matching process will be based either on individual masking and queue or on masking scheme with RXMGMASK, RX14MASK and RX15MASK, RXFGMASK. This bit can be written only in Freeze mode because it is blocked by hardware in other modes. 0 Individual Rx masking and queue feature are disabled. For backward compatibility with legacy applications, the reading of C/S word locks the MB even if it is EMPTY. 1 Individual Rx masking and queue feature are enabled.
16 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
17 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
18 LPRIOEN	Local Priority Enable This bit is provided for backwards compatibility with legacy applications. It controls whether the local priority feature is enabled or not. It is used to expand the ID used during the arbitration process. With this expanded ID concept, the arbitration process is done based on the full 32-bit word, but the actual transmitted ID still has 11-bit for standard frames and 29-bit for extended frames. This bit can be written only in Freeze mode because it is blocked by hardware in other modes. 0 Local Priority disabled. 1 Local Priority enabled.
19 AEN	Abort Enable This bit is supplied for backwards compatibility with legacy applications. When asserted, it enables the Tx abort mechanism. This mechanism guarantees a safe procedure for aborting a pending transmission, so that no frame is sent in the CAN bus without notification. This bit can be written only in Freeze mode because it is blocked by hardware in other modes. NOTE: When MCR[AEN] is asserted, only the abort mechanism (see Section "Transmission Abort Mechanism") must be used for updating Mailboxes configured for transmission. CAUTION: Writing the Abort code into Rx Mailboxes can cause unpredictable results when the MCR[AEN] is asserted. 0 Abort disabled. 1 Abort enabled.
20–21 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

Table continues on the next page...

CAN_MCR field descriptions (continued)

Field	Description
22–23 IDAM	ID Acceptance Mode This 2-bit field identifies the format of the Rx FIFO ID Filter Table elements. Note that all elements of the table are configured at the same time by this field (they are all the same format). See Section "Rx FIFO Structure". This field can be written only in Freeze mode because it is blocked by hardware in other modes. 00 Format A: One full ID (standard and extended) per ID Filter Table element. 01 Format B: Two full standard IDs or two partial 14-bit (standard and extended) IDs per ID Filter Table element. 10 Format C: Four partial 8-bit Standard IDs per ID Filter Table element. 11 Format D: All frames rejected.
24 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
25–31 MAXMB	Number Of The Last Message Buffer This 7-bit field defines the number of the last Message Buffers that will take part in the matching and arbitration processes. The reset value (0x0F) is equivalent to a 16 MB configuration. This field can be written only in Freeze mode because it is blocked by hardware in other modes. Number of the last MB = MAXMB NOTE: MAXMB must be programmed with a value smaller than the parameter NUMBER_OF_MB, otherwise the number of the last effective Message Buffer will be: (NUMBER_OF_MB - 1) Additionally, the value of MAXMB must encompass the FIFO size defined by CTRL2[RFFN]. MAXMB also impacts the definition of the minimum number of peripheral clocks per CAN bit as described in Table "Minimum Ratio Between Peripheral Clock Frequency and CAN Bit Rate" (in Section "Arbitration and Matching Timing").

40.4.3 Control 1 register (CAN_CTRL1)

This register is defined for specific FlexCAN control features related to the CAN bus, such as bit-rate, programmable sampling point within an Rx bit, Loop Back mode, Listen-Only mode, Bus Off recovery behavior and interrupt enabling (Bus-Off, Error, Warning). It also determines the Division Factor for the clock prescaler.

Address: 0h base + 4h offset = 4h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	PRES DIV								RJW		PSEG1			PSEG2		
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	BOFFMSK	ERRMSK	CLKSRC	LPB	TWRNMSK	RWRNMSK	0		SMP	BOFFREC	TSYN	LBUF	LOM	PROPSEG		
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

CAN_CTRL1 field descriptions

Field	Description
0–7 PRES DIV	Prescaler Division Factor This 8-bit field defines the ratio between the PE clock frequency and the Serial Clock (Sclock) frequency. The Sclock period defines the time quantum of the CAN protocol. For the reset value, the Sclock frequency is equal to the PE clock frequency. The Maximum value of this field is 0xFF, that gives a minimum Sclock frequency equal to the PE clock frequency divided by 256. See Section "Protocol Timing". This field can be written only in Freeze mode because it is blocked by hardware in other modes. Sclock frequency = PE clock frequency / (PRES DIV + 1)
8–9 RJW	Resync Jump Width This 2-bit field defines the maximum number of time quanta that a bit time can be changed by one re-synchronization. One time quantum is equal to the Sclock period. The valid programmable values are 0–3. This field can be written only in Freeze mode because it is blocked by hardware in other modes. Resync Jump Width = RJW + 1.
10–12 PSEG1	Phase Segment 1 This 3-bit field defines the length of Phase Buffer Segment 1 in the bit time. The valid programmable values are 0–7. This field can be written only in Freeze mode because it is blocked by hardware in other modes. Phase Buffer Segment 1 = (PSEG1 + 1) × Time-Quanta.
13–15 PSEG2	Phase Segment 2

Table continues on the next page...

CAN_CTRL1 field descriptions (continued)

Field	Description
	This 3-bit field defines the length of Phase Buffer Segment 2 in the bit time. The valid programmable values are 1–7. This field can be written only in Freeze mode because it is blocked by hardware in other modes. Phase Buffer Segment 2 = (PSEG2 + 1) × Time-Quanta.
16 BOFFMSK	Bus Off Mask This bit provides a mask for the Bus Off Interrupt. 0 Bus Off interrupt disabled. 1 Bus Off interrupt enabled.
17 ERRMSK	Error Mask This bit provides a mask for the Error Interrupt. 0 Error interrupt disabled. 1 Error interrupt enabled.
18 CLKSRC	CAN Engine Clock Source This bit selects the clock source to the CAN Protocol Engine (PE) to be either the peripheral clock (driven by the PLL) or the crystal oscillator clock. The selected clock is the one fed to the prescaler to generate the Serial Clock (Sclck). In order to guarantee reliable operation, this bit can be written only in Disable mode because it is blocked by hardware in other modes. See Section "Protocol Timing". 0 The CAN engine clock source is the oscillator clock. Under this condition, the oscillator clock frequency must be lower than the bus clock. 1 The CAN engine clock source is the peripheral clock.
19 LPB	Loop Back Mode This bit configures FlexCAN to operate in Loop-Back mode. In this mode, FlexCAN performs an internal loop back that can be used for self test operation. The bit stream output of the transmitter is fed back internally to the receiver input. The Rx CAN input pin is ignored and the Tx CAN output goes to the recessive state (logic 1). FlexCAN behaves as it normally does when transmitting, and treats its own transmitted message as a message received from a remote node. In this mode, FlexCAN ignores the bit sent during the ACK slot in the CAN frame acknowledge field, generating an internal acknowledge bit to ensure proper reception of its own message. Both transmit and receive interrupts are generated. This bit can be written only in Freeze mode because it is blocked by hardware in other modes. NOTE: In this mode, the MCR[SRXDIS] cannot be asserted because this will impede the self reception of a transmitted message. 0 Loop Back disabled. 1 Loop Back enabled.
20 TWRNMSK	Tx Warning Interrupt Mask This bit provides a mask for the Tx Warning Interrupt associated with the TWRNINT flag in the Error and Status Register. This bit is read as zero when MCR[WRNEN] bit is negated. This bit can be written only if MCR[WRNEN] bit is asserted. 0 Tx Warning Interrupt disabled. 1 Tx Warning Interrupt enabled.
21 RWRNMSK	Rx Warning Interrupt Mask

Table continues on the next page...

CAN_CTRL1 field descriptions (continued)

Field	Description
	This bit provides a mask for the Rx Warning Interrupt associated with the RWRNINT flag in the Error and Status Register. This bit is read as zero when MCR[WRNEN] bit is negated. This bit can be written only if MCR[WRNEN] bit is asserted. 0 Rx Warning Interrupt disabled. 1 Rx Warning Interrupt enabled.
22–23 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
24 SMP	CAN Bit Sampling This bit defines the sampling mode of CAN bits at the Rx input. This bit can be written only in Freeze mode because it is blocked by hardware in other modes. 0 Just one sample is used to determine the bit value. 1 Three samples are used to determine the value of the received bit: the regular one (sample point) and 2 preceding samples; a majority rule is used.
25 BOFFREC	Bus Off Recovery This bit defines how FlexCAN recovers from Bus Off state. If this bit is negated, automatic recovering from Bus Off state occurs according to the CAN Specification 2.0B. If the bit is asserted, automatic recovering from Bus Off is disabled and the module remains in Bus Off state until the bit is negated by the user. If the negation occurs before 128 sequences of 11 recessive bits are detected on the CAN bus, then Bus Off recovery happens as if the BOFFREC bit had never been asserted. If the negation occurs after 128 sequences of 11 recessive bits occurred, then FlexCAN will re-synchronize to the bus by waiting for 11 recessive bits before joining the bus. After negation, the BOFFREC bit can be re-asserted again during Bus Off, but it will be effective only the next time the module enters Bus Off. If BOFFREC was negated when the module entered Bus Off, asserting it during Bus Off will not be effective for the current Bus Off recovery. 0 Automatic recovering from Bus Off state enabled, according to CAN Spec 2.0 part B. 1 Automatic recovering from Bus Off state disabled.
26 TSYN	Timer Sync This bit enables a mechanism that resets the free-running timer each time a message is received in Message Buffer 0. This feature provides means to synchronize multiple FlexCAN stations with a special “SYNC” message, that is, global network time. If the RFEN bit in MCR is set (Rx FIFO enabled), the first available Mailbox, according to CTRL2[RFFN] setting, is used for timer synchronization instead of MB0. This bit can be written only in Freeze mode because it is blocked by hardware in other modes. 0 Timer Sync feature disabled 1 Timer Sync feature enabled
27 LBUF	Lowest Buffer Transmitted First This bit defines the ordering mechanism for Message Buffer transmission. When asserted, the LPRIEN bit does not affect the priority arbitration. This bit can be written only in Freeze mode because it is blocked by hardware in other modes. 0 Buffer with highest priority is transmitted first. 1 Lowest number buffer is transmitted first.
28 LOM	Listen-Only Mode This bit configures FlexCAN to operate in Listen-Only mode. In this mode, transmission is disabled, all error counters are frozen and the module operates in a CAN Error Passive mode. Only messages

Table continues on the next page...

CAN_CTRL1 field descriptions (continued)

Field	Description
	acknowledged by another CAN station will be received. If FlexCAN detects a message that has not been acknowledged, it will flag a BIT0 error without changing the REC, as if it was trying to acknowledge the message. Listen-Only mode acknowledgement can be obtained by the state of ESR1[FLTCNF] field which is Passive Error when Listen-Only mode is entered. There can be some delay between the Listen-Only mode request and acknowledge. This bit can be written only in Freeze mode because it is blocked by hardware in other modes. 0 Listen-Only mode is deactivated. 1 FlexCAN module operates in Listen-Only mode.
29–31 PROPSEG	Propagation Segment This 3-bit field defines the length of the Propagation Segment in the bit time. The valid programmable values are 0–7. This field can be written only in Freeze mode because it is blocked by hardware in other modes. Propagation Segment Time = (PROPSEG + 1) × Time-Quanta. Time-Quantum = one Sclock period.

40.4.4 Free Running Timer (CAN_TIMER)

This register represents a 16-bit free running counter that can be read and written by the CPU. The timer starts from 0x0 after Reset, counts linearly to 0xFFFF, and wraps around.

The timer is clocked by the FlexCAN bit-clock, which defines the baud rate on the CAN bus. During a message transmission/reception, it increments by one for each bit that is received or transmitted. When there is no message on the bus, it counts using the previously programmed baud rate. The timer is not incremented during Disable, Doze, Stop, and Freeze modes.

The timer value is captured when the second bit of the identifier field of any frame is on the CAN bus. This captured value is written into the Time Stamp entry in a message buffer after a successful reception or transmission of a message.

If bit CTRL1[TSYN] is asserted, the Timer is reset whenever a message is received in the first available Mailbox, according to CTRL2[RFFN] setting.

The CPU can write to this register anytime. However, if the write occurs at the same time that the Timer is being reset by a reception in the first Mailbox, then the write value is discarded.

Reading this register affects the Mailbox Unlocking procedure; see Section "Mailbox Lock Mechanism".

Memory map/register definition

Address: 0h base + 8h offset = 8h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0																TIMER															
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	

CAN_TIMER field descriptions

Field	Description
0–15 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
16–31 TIMER	Timer Value Contains the free-running counter value.

40.4.5 Rx Mailboxes Global Mask Register (CAN_RXMGMASK)

This register is located in RAM.

RXMGMASK is provided for legacy application support.

- When the MCR[IRMQ] bit is negated, RXMGMASK is always in effect.
- When the MCR[IRMQ] bit is asserted, RXMGMASK has no effect.

RXMGMASK is used to mask the filter fields of all Rx MBs, excluding MBs 14-15, which have individual mask registers.

This register can only be written in Freeze mode as it is blocked by hardware in other modes.

Address: 0h base + 10h offset = 10h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R																																
W	MG[31:0]																															
Reset	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	

CAN_RXMGMASK field descriptions

Field	Description
0–31 MG[31:0]	Rx Mailboxes Global Mask Bits These bits mask the Mailbox filter bits. Note that the alignment with the ID word of the Mailbox is not perfect as the two most significant MG bits affect the fields RTR and IDE, which are located in the Control and Status word of the Mailbox. The following table shows in detail which MG bits mask each Mailbox filter field.

CAN_RXMGMASK field descriptions (continued)

Field	Description						
	SMB[RTR] ¹	CTRL2[RRS]	CTRL2[EACEN]	Mailbox filter fields			
				MB[RTR]	MB[IDE]	MB[ID]	Reserved
	0	-	0	note ²	note ³	MG[28:0]	MG[31:29]
	0	-	1	MG[31]	MG[30]	MG[28:0]	MG[29]
	1	0	-	-	-	-	MG[31:0]
	1	1	0	-	-	MG[28:0]	MG[31:29]
	1	1	1	MG[31]	MG[30]	MG[28:0]	MG[29]
1. RTR bit of the Incoming Frame. It is saved into an auxiliary MB called Rx Serial Message Buffer (Rx SMB). 2. If the CTRL2[EACEN] bit is negated, the RTR bit of Mailbox is never compared with the RTR bit of the incoming frame. 3. If the CTRL2[EACEN] bit is negated, the IDE bit of Mailbox is always compared with the IDE bit of the incoming frame. 0 The corresponding bit in the filter is "don't care." 1 The corresponding bit in the filter is checked.							

1. RTR bit of the Incoming Frame. It is saved into an auxiliary MB called Rx Serial Message Buffer (Rx SMB).
2. If the CTRL2[EACEN] bit is negated, the RTR bit of Mailbox is never compared with the RTR bit of the incoming frame.
3. If the CTRL2[EACEN] bit is negated, the IDE bit of Mailbox is always compared with the IDE bit of the incoming frame.

40.4.6 Rx 14 Mask register (CAN_RX14MASK)

This register is located in RAM.

RX14MASK is provided for legacy application support. When the MCR[IRMQ] bit is asserted, RX14MASK has no effect.

RX14MASK is used to mask the filter fields of Message Buffer 14.

This register can only be programmed while the module is in Freeze mode as it is blocked by hardware in other modes.

Address: 0h base + 14h offset = 14h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	RX14M[31:0]																															
W																																
Reset	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	

CAN_RX14MASK field descriptions

Field	Description
0–31 RX14M[31:0]	Rx Buffer 14 Mask Bits Each mask bit masks the corresponding Mailbox 14 filter field in the same way that RXMGMASK masks other Mailboxes' filters. See the description of the CAN_RXMGMASK register.

CAN_RX14MASK field descriptions (continued)

Field	Description
0	The corresponding bit in the filter is "don't care."
1	The corresponding bit in the filter is checked.

40.4.7 Rx 15 Mask register (CAN_RX15MASK)

This register is located in RAM.

RX15MASK is provided for legacy application support. When the MCR[IRMQ] bit is asserted, RX15MASK has no effect.

RX15MASK is used to mask the filter fields of Message Buffer 15.

This register can be programmed only while the module is in Freeze mode because it is blocked by hardware in other modes.

Address: 0h base + 18h offset = 18h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R																																
W																																
	RX15M[31:0]																															
Reset	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	

CAN_RX15MASK field descriptions

Field	Description
0–31 RX15M[31:0]	Rx Buffer 15 Mask Bits Each mask bit masks the corresponding Mailbox 15 filter field in the same way that RXMGMASK masks other Mailboxes' filters. See the description of the CAN_RXMGMASK register. 0 The corresponding bit in the filter is "don't care." 1 The corresponding bit in the filter is checked.

40.4.8 Error Counter (CAN_ECR)

This register has two 8-bit fields reflecting the value of two FlexCAN error counters: Transmit Error Counter (TXERRCNT field) and Receive Error Counter (RXERRCNT field). The rules for increasing and decreasing these counters are described in the CAN protocol and are completely implemented in the FlexCAN module. Both counters are read-only except in Freeze mode, where they can be written by the CPU.

FlexCAN responds to any bus state as described in the protocol, for example, transmit Error Active or Error Passive flag, delay its transmission start time (Error Passive) and avoid any influence on the bus when in Bus Off state.

The following are the basic rules for FlexCAN bus state transitions:

- If the value of TXERRCNT or RXERRCNT increases to be greater than or equal to 128, the FLTCONF field in the Error and Status Register is updated to reflect 'Error Passive' state.
- If the FlexCAN state is 'Error Passive', and either TXERRCNT or RXERRCNT decrements to a value less than or equal to 127 while the other already satisfies this condition, the FLTCONF field in the Error and Status Register is updated to reflect 'Error Active' state.
- If the value of TXERRCNT increases to be greater than 255, the FLTCONF field in the Error and Status Register is updated to reflect 'Bus Off' state, and an interrupt may be issued. The value of TXERRCNT is then reset to zero.
- If FlexCAN is in 'Bus Off' state, then TXERRCNT is cascaded together with another internal counter to count the 128th occurrences of 11 consecutive recessive bits on the bus. Hence, TXERRCNT is reset to zero and counts in a manner where the internal counter counts 11 such bits and then wraps around while incrementing the TXERRCNT. When TXERRCNT reaches the value of 128, the FLTCONF field in the Error and Status Register is updated to be 'Error Active' and both error counters are reset to zero. At any instance of dominant bit following a stream of less than 11 consecutive recessive bits, the internal counter resets itself to zero without affecting the TXERRCNT value.
- If during system start-up, only one node is operating, then its TXERRCNT increases in each message it is trying to transmit, as a result of acknowledge errors (indicated by the ACKERR bit in the Error and Status Register). After the transition to 'Error Passive' state, the TXERRCNT does not increment anymore by acknowledge errors. Therefore the device never goes to the 'Bus Off' state.
- If the RXERRCNT increases to a value greater than 127, it is not incremented further, even if more errors are detected while being a receiver. At the next successful message reception, the counter is set to a value between 119 and 127 to resume to 'Error Active' state.

Address: 0h base + 1Ch offset = 1Ch

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0																RXERRCNT							TXERRCNT								
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	

CAN_ECR field descriptions

Field	Description
0–15 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
16–23 RXERRCNT	Receive Error Counter

Table continues on the next page...

CAN_ECR field descriptions (continued)

Field	Description
24–31 TXERRCNT	Transmit Error Counter

40.4.9 Error and Status 1 register (CAN_ESR1)

This register reflects various error conditions and some general status of the device. It is also the source of interrupts to the CPU.

A CPU read operation clears the following fields to 0, so these fields report error conditions that occurred after the last time the CPU read this register: BIT1ERR, BIT0ERR, ACKERR, CRCERR, FRMERR, and STFERR. TXWRN, RXWRN, IDLE, TX, FLTCNF, and RX provide status information.

The following table shows the FlexCAN state variables and their meanings. Combinations not shown in the table are reserved.

SYNCH	IDLE	TX	RX	FlexCAN state
0	0	0	0	Not synchronized to CAN bus
1	1	x	x	Idle
1	0	1	0	Transmitting
1	0	0	1	Receiving

Address: 0h base + 20h offset = 20h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0												SYNCH	TWRNINT	RWRNINT	
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	BIT1ERR	BIT0ERR	ACKERR	CRCERR	FRMERR	STFERR	TXWRN	RXWRN	IDLE	TX	FLTCONF		RX	BOFFINT	ERRINT	0
W														w1c	w1c	
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

CAN_ESR1 field descriptions

Field	Description
0–12 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
13 SYNCH	CAN Synchronization Status This read-only flag indicates whether the FlexCAN is synchronized to the CAN bus and able to participate in the communication process. It is set and cleared by the FlexCAN. See the table in the overall CAN_ESR1 register description. 0 FlexCAN is not synchronized to the CAN bus. 1 FlexCAN is synchronized to the CAN bus.
14 TWRNINT	Tx Warning Interrupt Flag If the WRNEN bit in MCR is asserted, the TWRNINT bit is set when the TXWRN flag transitions from 0 to 1, meaning that the Tx error counter reached 96. If the corresponding mask bit in the Control Register (TWRNMSK) is set, an interrupt is generated to the CPU. This bit is cleared by writing it to 1. When WRNEN is negated, this flag is masked. CPU must clear this flag before disabling the bit. Otherwise it will be set when the WRNEN is set again. Writing 0 has no effect. This flag is not generated during Bus Off state. This bit is not updated during Freeze mode. 0 No such occurrence. 1 The Tx error counter transitioned from less than 96 to greater than or equal to 96.
15 RWRNINT	Rx Warning Interrupt Flag If the WRNEN bit in MCR is asserted, the RWRNINT bit is set when the RXWRN flag transitions from 0 to 1, meaning that the Rx error counters reached 96. If the corresponding mask bit in the Control Register (RWRNMSK) is set, an interrupt is generated to the CPU. This bit is cleared by writing it to 1. When WRNEN is negated, this flag is masked. CPU must clear this flag before disabling the bit. Otherwise it will be set when the WRNEN is set again. Writing 0 has no effect. This bit is not updated during Freeze mode. 0 No such occurrence. 1 The Rx error counter transitioned from less than 96 to greater than or equal to 96.
16 BIT1ERR	Bit1 Error This bit indicates when an inconsistency occurs between the transmitted and the received bit in a message.

Table continues on the next page...

CAN_ESR1 field descriptions (continued)

Field	Description
	NOTE: This bit is not set by a transmitter in case of arbitration field or ACK slot, or in case of a node sending a passive error flag that detects dominant bits. 0 No such occurrence. 1 At least one bit sent as recessive is received as dominant.
17 BIT0ERR	Bit0 Error This bit indicates when an inconsistency occurs between the transmitted and the received bit in a message. 0 No such occurrence. 1 At least one bit sent as dominant is received as recessive.
18 ACKERR	Acknowledge Error This bit indicates that an Acknowledge Error has been detected by the transmitter node, that is, a dominant bit has not been detected during the ACK SLOT. 0 No such occurrence. 1 An ACK error occurred since last read of this register.
19 CRCERR	Cyclic Redundancy Check Error This bit indicates that a CRC Error has been detected by the receiver node, that is, the calculated CRC is different from the received. 0 No such occurrence. 1 A CRC error occurred since last read of this register.
20 FRMERR	Form Error This bit indicates that a Form Error has been detected by the receiver node, that is, a fixed-form bit field contains at least one illegal bit. 0 No such occurrence. 1 A Form Error occurred since last read of this register.
21 STFERR	Stuffing Error This bit indicates that a Stuffing Error has been detected. 0 No such occurrence. 1 A Stuffing Error occurred since last read of this register.
22 TXWRN	TX Error Warning This bit indicates when repetitive errors are occurring during message transmission. This bit is not updated during Freeze mode. 0 No such occurrence. 1 TXERRCNT is greater than or equal to 96.
23 RXWRN	Rx Error Warning This bit indicates when repetitive errors are occurring during message reception. This bit is not updated during Freeze mode. 0 No such occurrence. 1 RXERRCNT is greater than or equal to 96.

Table continues on the next page...

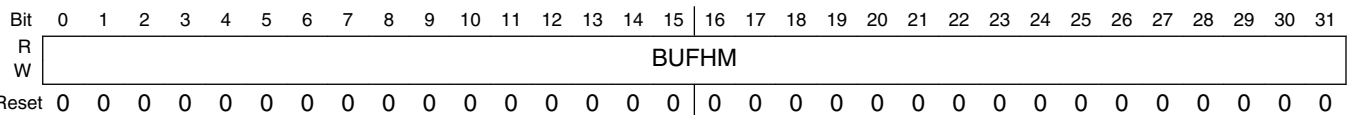
CAN_ESR1 field descriptions (continued)

Field	Description
24 IDLE	This bit indicates when CAN bus is in IDLE state. See the table in the overall CAN_ESR1 register description. 0 No such occurrence. 1 CAN bus is now IDLE.
25 TX	FlexCAN In Transmission This bit indicates if FlexCAN is transmitting a message. See the table in the overall CAN_ESR1 register description. 0 FlexCAN is not transmitting a message. 1 FlexCAN is transmitting a message.
26–27 FLTCONF	Fault Confinement State This 2-bit field indicates the Confinement State of the FlexCAN module. If the LOM bit in the Control Register is asserted, after some delay that depends on the CAN bit timing the FLTCONF field will indicate “Error Passive”. The very same delay affects the way how FLTCONF reflects an update to ECR register by the CPU. It may be necessary up to one CAN bit time to get them coherent again. Because the Control Register is not affected by soft reset, the FLTCONF field will not be affected by soft reset if the LOM bit is asserted. 00 Error Active 01 Error Passive 1x Bus Off
28 RX	FlexCAN In Reception This bit indicates if FlexCAN is receiving a message. See the table in the overall CAN_ESR1 register description. 0 FlexCAN is not receiving a message. 1 FlexCAN is receiving a message.
29 BOFFINT	Bus Off Interrupt This bit is set when FlexCAN enters ‘Bus Off’ state. If the corresponding mask bit in the Control Register (BOFFMSK) is set, an interrupt is generated to the CPU. This bit is cleared by writing it to 1. Writing 0 has no effect. 0 No such occurrence. 1 FlexCAN module entered Bus Off state.
30 ERRINT	Error Interrupt This bit indicates that at least one of the Error Bits (bits 15-10) is set. If the corresponding mask bit CTRL1[ERRMSK] is set, an interrupt is generated to the CPU. This bit is cleared by writing it to 1. Writing 0 has no effect. 0 No such occurrence. 1 Indicates setting of any Error Bit in the Error and Status Register.
31 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

40.4.10 Interrupt Masks 2 register (CAN_IMASK2)

This register allows any number of a range of the 32 Message Buffer Interrupts to be enabled or disabled for MB63 to MB32. It contains one interrupt mask bit per buffer, enabling the CPU to determine which buffer generates an interrupt after a successful transmission or reception, that is, when the corresponding IFLAG2 bit is set.

Address: 0h base + 24h offset = 24h



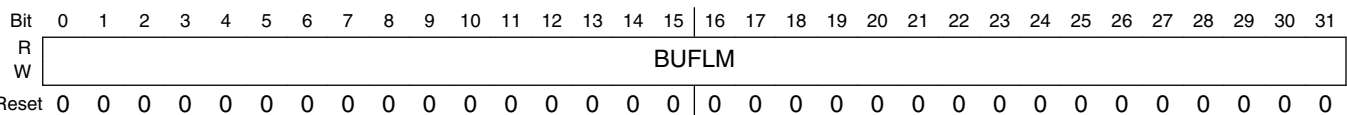
CAN_IMASK2 field descriptions

Field	Description
0–31 BUFHM	Buffer MB _i Mask Each bit enables or disables the corresponding FlexCAN Message Buffer Interrupt for MB63 to MB32. NOTE: Setting or clearing a bit in the IMASK2 Register can assert or negate an interrupt request, if the corresponding IFLAG2 bit is set. 0 The corresponding buffer Interrupt is disabled. 1 The corresponding buffer Interrupt is enabled.

40.4.11 Interrupt Masks 1 register (CAN_IMASK1)

This register allows any number of a range of the 32 Message Buffer Interrupts to be enabled or disabled for MB31 to MB0. It contains one interrupt mask bit per buffer, enabling the CPU to determine which buffer generates an interrupt after a successful transmission or reception, that is, when the corresponding IFLAG1 bit is set.

Address: 0h base + 28h offset = 28h



CAN_IMASK1 field descriptions

Field	Description
0–31 BUFLM	Buffer MB_i Mask Each bit enables or disables the corresponding FlexCAN Message Buffer Interrupt for MB31 to MB0. NOTE: Setting or clearing a bit in the IMASK1 Register can assert or negate an interrupt request, if the corresponding IFLAG1 bit is set. 0 The corresponding buffer Interrupt is disabled. 1 The corresponding buffer Interrupt is enabled.

40.4.12 Interrupt Flags 2 register (CAN_IFLAG2)

This register defines the flags for the 32 Message Buffer interrupts for MB63 to MB32. It contains one interrupt flag bit per buffer. Each successful transmission or reception sets the corresponding IFLAG2 bit. If the corresponding IMASK2 bit is set, an interrupt will be generated. The interrupt flag must be cleared by writing 1 to it. Writing 0 has no effect.

Before updating MCR[MAXMB] field, CPU must service the IFLAG2 bits whose MB value is greater than the MCR[MAXMB] to be updated; otherwise, they will remain set and be inconsistent with the number of MBs available.

Address: 0h base + 2Ch offset = 2Ch

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	BUFHI																															
W	w1c																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	

CAN_IFLAG2 field descriptions

Field	Description
0–31 BUFHI	Buffer MB_i Interrupt Each bit flags the corresponding FlexCAN Message Buffer interrupt for MB63 to MB32. 0 The corresponding buffer has no occurrence of successfully completed transmission or reception. 1 The corresponding buffer has successfully completed transmission or reception.

40.4.13 Interrupt Flags 1 register (CAN_IFLAG1)

This register defines the flags for the 32 Message Buffer interrupts for MB31 to MB0. It contains one interrupt flag bit per buffer. Each successful transmission or reception sets the corresponding IFLAG1 bit. If the corresponding IMASK1 bit is set, an interrupt will be generated. The interrupt flag must be cleared by writing 1 to it. Writing 0 has no effect.

The BUF7I to BUF5I flags are also used to represent FIFO interrupts when the Rx FIFO is enabled. When the bit MCR[RFEN] is set, the function of the 8 least significant interrupt flags BUF[7:0]I changes: BUF7I, BUF6I and BUF5I indicate operating conditions of the FIFO, and the BUF4TO0I field is reserved.

Before enabling the RFEN, the CPU must service the IFLAG bits asserted in the Rx FIFO region; see Section "Rx FIFO". Otherwise, these IFLAG bits will mistakenly show the related MBs now belonging to FIFO as having contents to be serviced. When the RFEN bit is negated, the FIFO flags must be cleared. The same care must be taken when an RFFN value is selected extending Rx FIFO filters beyond MB7. For example, when RFFN is 0x8, the MB0-23 range is occupied by Rx FIFO filters and related IFLAG bits must be cleared.

Before updating MCR[MAXMB] field, CPU must service the IFLAG1 bits whose MB value is greater than the MCR[MAXMB] to be updated; otherwise, they will remain set and be inconsistent with the number of MBs available.

Address: 0h base + 30h offset = 30h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	BUF31TO8I															
W	w1c															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	BUF31TO8I								BUF7I	BUF6I	BUF5I	BUF4TO1I				BUF0I
W	w1c								w1c	w1c	w1c	w1c				w1c
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

CAN_IFLAG1 field descriptions

Field	Description
0–23 BUF31TO8I	Buffer MB_i Interrupt Each bit flags the corresponding FlexCAN Message Buffer interrupt for MB31 to MB8. 0 The corresponding buffer has no occurrence of successfully completed transmission or reception. 1 The corresponding buffer has successfully completed transmission or reception.
24 BUF7I	Buffer MB7 Interrupt Or "Rx FIFO Overflow" When the RFEN bit in the MCR is cleared (Rx FIFO disabled), this bit flags the interrupt for MB7. NOTE: This flag is cleared by the FlexCAN whenever the bit MCR[RFEN] is changed by CPU writes. The BUF7I flag represents "Rx FIFO Overflow" when MCR[RFEN] is set. In this case, the flag indicates that a message was lost because the Rx FIFO is full. Note that the flag will not be asserted when the Rx FIFO is full and the message was captured by a Mailbox. 0 No occurrence of MB7 completing transmission/reception when MCR[RFEN]=0, or of Rx FIFO overflow when MCR[RFEN]=1 1 MB7 completed transmission/reception when MCR[RFEN]=0, or Rx FIFO overflow when MCR[RFEN]=1
25 BUF6I	Buffer MB6 Interrupt Or "Rx FIFO Warning" When the RFEN bit in the MCR is cleared (Rx FIFO disabled), this bit flags the interrupt for MB6. NOTE: This flag is cleared by the FlexCAN whenever the bit MCR[RFEN] is changed by CPU writes. The BUF6I flag represents "Rx FIFO Warning" when MCR[RFEN] is set. In this case, the flag indicates when the number of unread messages within the Rx FIFO is increased to 5 from 4 due to the reception of a new one, meaning that the Rx FIFO is almost full. Note that if the flag is cleared while the number of unread messages is greater than 4, it does not assert again until the number of unread messages within the Rx FIFO is decreased to be equal to or less than 4. 0 No occurrence of MB6 completing transmission/reception when MCR[RFEN]=0, or of Rx FIFO almost full when MCR[RFEN]=1 1 MB6 completed transmission/reception when MCR[RFEN]=0, or Rx FIFO almost full when MCR[RFEN]=1
26 BUF5I	Buffer MB5 Interrupt Or "Frames available in Rx FIFO" When the RFEN bit in the MCR is cleared (Rx FIFO disabled), this bit flags the interrupt for MB5. NOTE: This flag is cleared by the FlexCAN whenever the bit MCR[RFEN] is changed by CPU writes. The BUF5I flag represents "Frames available in Rx FIFO" when MCR[RFEN] is set. In this case, the flag indicates that at least one frame is available to be read from the Rx FIFO. 0 No occurrence of MB5 completing transmission/reception when MCR[RFEN]=0, or of frame(s) available in the FIFO, when MCR[RFEN]=1 1 MB5 completed transmission/reception when MCR[RFEN]=0, or frame(s) available in the Rx FIFO when MCR[RFEN]=1
27–30 BUF4TO1I	Buffer MB_i Interrupt Or "reserved" When the RFEN bit in the MCR is cleared (Rx FIFO disabled), these bits flag the interrupts for MB4 to MB1. NOTE: These flags are cleared by the FlexCAN whenever the bit MCR[RFEN] is changed by CPU writes. The BUF4TO1I flags are reserved when MCR[RFEN] is set.

Table continues on the next page...

CAN_IFLAG1 field descriptions (continued)

Field	Description
	0 The corresponding buffer has no occurrence of successfully completed transmission or reception when MCR[RFEN]=0. 1 The corresponding buffer has successfully completed transmission or reception when MCR[RFEN]=0.
31 BUF0I	Buffer MB0 Interrupt Or "reserved" When the RFEN bit in the MCR is cleared (Rx FIFO disabled), this bit flags the interrupt for MB0. NOTE: This flag is cleared by the FlexCAN whenever the bit MCR[RFEN] is changed by CPU writes. The BUF0I flag is reserved when MCR[RFEN] is set. 0 The corresponding buffer has no occurrence of successfully completed transmission or reception when MCR[RFEN]=0. 1 The corresponding buffer has successfully completed transmission or reception when MCR[RFEN]=0.

40.4.14 Control 2 register (CAN_CTRL2)

This register contains control bits for CAN errors, FIFO features, and mode selection.

Address: 0h base + 34h offset = 34h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0		ECRWRE	WRMFRZ												
W																
Reset	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0
Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R																
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

CAN_CTRL2 field descriptions

Field	Description
0–1 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
2 ECRWRE	Error-correction Configuration Register Write Enable Enable that the MECCR register is updated. This bit is automatically set to 0 if the protocol described in the "Detection and Correction of Memory Errors" section is not followed. 0 Disable update. 1 Enable update.

Table continues on the next page...

CAN_CTRL2 field descriptions (continued)

Field	Description																																																																																										
3 WRMFRZ	Write-Access To Memory In Freeze Mode Enable unrestricted write access to FlexCAN memory in Freeze mode. This bit can only be written in Freeze mode and has no effect out of Freeze mode. 0 Maintain the write access restrictions. 1 Enable unrestricted write access to FlexCAN memory.																																																																																										
4–7 RFFN	Number Of Rx FIFO Filters This 4-bit field defines the number of Rx FIFO filters, as shown in the following table. The maximum selectable number of filters is determined by the chip. This field can only be written in Freeze mode as it is blocked by hardware in other modes. This field must not be programmed with values that make the number of Message Buffers occupied by Rx FIFO and ID Filter exceed the number of Mailboxes present, defined by MCR[MAXMB]. NOTE: Each group of eight filters occupies a memory space equivalent to two Message Buffers which means that the more filters are implemented the less Mailboxes will be available. Considering that the Rx FIFO occupies the memory space originally reserved for MB0-5, RFFN should be programmed with a value corresponding to a number of filters not greater than the number of available memory words which can be calculated as follows: $(\text{SETUP\_MB} - 6) \times 4$ where SETUP_MB is the least between NUMBER_OF_MB and MAXMB. The number of remaining Mailboxes available will be: $(\text{SETUP\_MB} - 8) - (\text{RFFN} \times 2)$ If the Number of Rx FIFO Filters programmed through RFFN exceeds the SETUP_MB value (memory space available) the exceeding ones will not be functional. <table><tr><th>RFFN[3:0]</th><th>Number of Rx FIFO filters</th><th>Message Buffers occupied by Rx FIFO and ID Filter Table</th><th>Remaining Available Mailboxes¹</th><th>Rx FIFO ID Filter Table Elements Affected by Rx Individual Masks²</th><th>Rx FIFO ID Filter Table Elements Affected by Rx FIFO Global Mask²</th></tr><tr><td>0x0</td><td>8</td><td>MB 0-7</td><td>MB 8-63</td><td>Elements 0-7</td><td>none</td></tr><tr><td>0x1</td><td>16</td><td>MB 0-9</td><td>MB 10-63</td><td>Elements 0-9</td><td>Elements 10-15</td></tr><tr><td>0x2</td><td>24</td><td>MB 0-11</td><td>MB 12-63</td><td>Elements 0-11</td><td>Elements 12-23</td></tr><tr><td>0x3</td><td>32</td><td>MB 0-13</td><td>MB 14-63</td><td>Elements 0-13</td><td>Elements 14-31</td></tr><tr><td>0x4</td><td>40</td><td>MB 0-15</td><td>MB 16-63</td><td>Elements 0-15</td><td>Elements 16-39</td></tr><tr><td>0x5</td><td>48</td><td>MB 0-17</td><td>MB 18-63</td><td>Elements 0-17</td><td>Elements 18-47</td></tr><tr><td>0x6</td><td>56</td><td>MB 0-19</td><td>MB 20-63</td><td>Elements 0-19</td><td>Elements 20-55</td></tr><tr><td>0x7</td><td>64</td><td>MB 0-21</td><td>MB 22-63</td><td>Elements 0-21</td><td>Elements 22-63</td></tr><tr><td>0x8</td><td>72</td><td>MB 0-23</td><td>MB 24-63</td><td>Elements 0-23</td><td>Elements 24-71</td></tr><tr><td>0x9</td><td>80</td><td>MB 0-25</td><td>MB 26-63</td><td>Elements 0-25</td><td>Elements 26-79</td></tr><tr><td>0xA</td><td>88</td><td>MB 0-27</td><td>MB 28-63</td><td>Elements 0-27</td><td>Elements 28-87</td></tr><tr><td>0xB</td><td>96</td><td>MB 0-29</td><td>MB 30-63</td><td>Elements 0-29</td><td>Elements 30-95</td></tr><tr><td>0xC</td><td>104</td><td>MB 0-31</td><td>MB 32-63</td><td>Elements 0-31</td><td>Elements 32-103</td></tr><tr><td>0xD</td><td>112</td><td>MB 0-33</td><td>MB 34-63</td><td>Elements 0-31</td><td>Elements 32-111</td></tr></table>	RFFN[3:0]	Number of Rx FIFO filters	Message Buffers occupied by Rx FIFO and ID Filter Table	Remaining Available Mailboxes ¹	Rx FIFO ID Filter Table Elements Affected by Rx Individual Masks ²	Rx FIFO ID Filter Table Elements Affected by Rx FIFO Global Mask ²	0x0	8	MB 0-7	MB 8-63	Elements 0-7	none	0x1	16	MB 0-9	MB 10-63	Elements 0-9	Elements 10-15	0x2	24	MB 0-11	MB 12-63	Elements 0-11	Elements 12-23	0x3	32	MB 0-13	MB 14-63	Elements 0-13	Elements 14-31	0x4	40	MB 0-15	MB 16-63	Elements 0-15	Elements 16-39	0x5	48	MB 0-17	MB 18-63	Elements 0-17	Elements 18-47	0x6	56	MB 0-19	MB 20-63	Elements 0-19	Elements 20-55	0x7	64	MB 0-21	MB 22-63	Elements 0-21	Elements 22-63	0x8	72	MB 0-23	MB 24-63	Elements 0-23	Elements 24-71	0x9	80	MB 0-25	MB 26-63	Elements 0-25	Elements 26-79	0xA	88	MB 0-27	MB 28-63	Elements 0-27	Elements 28-87	0xB	96	MB 0-29	MB 30-63	Elements 0-29	Elements 30-95	0xC	104	MB 0-31	MB 32-63	Elements 0-31	Elements 32-103	0xD	112	MB 0-33	MB 34-63	Elements 0-31	Elements 32-111
RFFN[3:0]	Number of Rx FIFO filters	Message Buffers occupied by Rx FIFO and ID Filter Table	Remaining Available Mailboxes ¹	Rx FIFO ID Filter Table Elements Affected by Rx Individual Masks ²	Rx FIFO ID Filter Table Elements Affected by Rx FIFO Global Mask ²																																																																																						
0x0	8	MB 0-7	MB 8-63	Elements 0-7	none																																																																																						
0x1	16	MB 0-9	MB 10-63	Elements 0-9	Elements 10-15																																																																																						
0x2	24	MB 0-11	MB 12-63	Elements 0-11	Elements 12-23																																																																																						
0x3	32	MB 0-13	MB 14-63	Elements 0-13	Elements 14-31																																																																																						
0x4	40	MB 0-15	MB 16-63	Elements 0-15	Elements 16-39																																																																																						
0x5	48	MB 0-17	MB 18-63	Elements 0-17	Elements 18-47																																																																																						
0x6	56	MB 0-19	MB 20-63	Elements 0-19	Elements 20-55																																																																																						
0x7	64	MB 0-21	MB 22-63	Elements 0-21	Elements 22-63																																																																																						
0x8	72	MB 0-23	MB 24-63	Elements 0-23	Elements 24-71																																																																																						
0x9	80	MB 0-25	MB 26-63	Elements 0-25	Elements 26-79																																																																																						
0xA	88	MB 0-27	MB 28-63	Elements 0-27	Elements 28-87																																																																																						
0xB	96	MB 0-29	MB 30-63	Elements 0-29	Elements 30-95																																																																																						
0xC	104	MB 0-31	MB 32-63	Elements 0-31	Elements 32-103																																																																																						
0xD	112	MB 0-33	MB 34-63	Elements 0-31	Elements 32-111																																																																																						

Table continues on the next page...

CAN_CTRL2 field descriptions (continued)

Field	Description					
	RFFN[3:0]	Number of Rx FIFO filters	Message Buffers occupied by Rx FIFO and ID Filter Table	Remaining Available Mailboxes ¹	Rx FIFO ID Filter Table Elements Affected by Rx Individual Masks ²	Rx FIFO ID Filter Table Elements Affected by Rx FIFO Global Mask ²
	0xE	120	MB 0-35	MB 36-63	Elements 0-31	Elements 32-119
	0xF	128	MB 0-37	MB 38-63	Elements 0-31	Elements 32-127
	1. The number of the last remaining available mailboxes is defined by the least value between the parameter NUMBER_OF_MB minus 1 and the MCR[MAXMB] field. 2. If Rx Individual Mask Registers are not enabled then all Rx FIFO filters are affected by the Rx FIFO Global Mask.					
8–12 TASD	Tx Arbitration Start Delay This 5-bit field indicates how many CAN bits the Tx arbitration process start point can be delayed from the first bit of CRC field on CAN bus. This field can be written only in Freeze mode because it is blocked by hardware in other modes. This field is useful to optimize the transmit performance based on factors such as: peripheral/serial clock ratio, CAN bit timing and number of MBs. The duration of an arbitration process, in terms of CAN bits, is directly proportional to the number of available MBs and CAN baud rate and inversely proportional to the peripheral clock frequency. The optimal arbitration timing is that in which the last MB is scanned right before the first bit of the Intermission field of a CAN frame. Therefore, if there are few MBs and the system/serial clock ratio is high and the CAN baud rate is low then the arbitration can be delayed and vice-versa. If TASD is 0 then the arbitration start is not delayed, thus the CPU has less time to configure a Tx MB for the next arbitration, but more time is reserved for arbitration. On the other hand, if TASD is 24 then the CPU can configure a Tx MB later and less time is reserved for arbitration. If too little time is reserved for arbitration the FlexCAN may be not able to find winner MBs in time to compete with other nodes for the CAN bus. If the arbitration ends too much time before the first bit of Intermission field then there is a chance that the CPU reconfigures some Tx MBs and the winner MB is not the best to be transmitted. The optimal configuration for TASD can be calculated as: $TASD = 25 - \left\{ f_{CANCLK} \times [MAXMB + 3 - (RFEN \times 8) - (RFEN \times RFFN \times 2)] \times 2 \right\} / \left\{ f_{SYS} \times [1 + (PSEG1+1) + (PSEG2+1) + (PROPSEG+1)] \times (PRES DIV+1) \right\}$ where: • f_{CANCLK} is the Protocol Engine (PE) Clock (see section "Protocol Timing"), in Hz • f_{SYS} is the peripheral clock, in Hz • MAXMB is the value in CTRL1[MAXMB] field • RFEN is the value in CTRL1[RFEN] bit • RFFN is the value in CTRL2[RFFN] field • PSEG1 is the value in CTRL1[PSEG1] field • PSEG2 is the value in CTRL1[PSEG2] field • PROPSEG is the value in CTRL1[PROPSEG] field • PRES DIV is the value in CTRL1[PRES DIV] field					

Table continues on the next page...

CAN_CTRL2 field descriptions (continued)

Field	Description
	See Section "Arbitration process" and Section "Protocol Timing" for more details.
13 MRP	Mailboxes Reception Priority If this bit is set the matching process starts from the Mailboxes and if no match occurs the matching continues on the Rx FIFO. This bit can be written only in Freeze mode because it is blocked by hardware in other modes. 0 Matching starts from Rx FIFO and continues on Mailboxes. 1 Matching starts from Mailboxes and continues on Rx FIFO.
14 RRS	Remote Request Storing If this bit is asserted Remote Request Frame is submitted to a matching process and stored in the corresponding Message Buffer in the same fashion of a Data Frame. No automatic Remote Response Frame will be generated. If this bit is negated the Remote Request Frame is submitted to a matching process and an automatic Remote Response Frame is generated if a Message Buffer with CODE=0b1010 is found with the same ID. This bit can be written only in Freeze mode because it is blocked by hardware in other modes. 0 Remote Response Frame is generated. 1 Remote Request Frame is stored.
15 EACEN	Entire Frame Arbitration Field Comparison Enable For Rx Mailboxes This bit controls the comparison of IDE and RTR bits within Rx Mailboxes filters with their corresponding bits in the incoming frame by the matching process. This bit does not affect matching for Rx FIFO. This bit can be written only in Freeze mode because it is blocked by hardware in other modes. 0 Rx Mailbox filter's IDE bit is always compared and RTR is never compared despite mask bits. 1 Enables the comparison of both Rx Mailbox filter's IDE and RTR bit with their corresponding bits within the incoming frame. Mask bits do apply.
16–31 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

1. The number of the last remaining available mailboxes is defined by the least value between the parameter NUMBER_OF_MB minus 1 and the MCR[MAXMB] field.
2. If Rx Individual Mask Registers are not enabled then all Rx FIFO filters are affected by the Rx FIFO Global Mask.

40.4.15 Error and Status 2 register (CAN_ESR2)

This register reflects various interrupt flags and some general status.

Address: 0h base + 38h offset = 38h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0								LPTM							
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	VPS	IMB	0												
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

CAN_ESR2 field descriptions

Field	Description
0–8 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
9–15 LPTM	Lowest Priority Tx Mailbox If ESR2[VPS] is asserted, this field indicates the lowest number inactive Mailbox (see the IMB bit description). If there is no inactive Mailbox then the Mailbox indicated depends on CTRL1[LBUF] bit value. If CTRL1[LBUF] bit is negated then the Mailbox indicated is the one that has the greatest arbitration value (see the "Highest priority Mailbox first" section). If CTRL1[LBUF] bit is asserted then the Mailbox indicated is the highest number active Tx Mailbox. If a Tx Mailbox is being transmitted it is not considered in LPTM calculation. If ESR2[IMB] is not asserted and a frame is transmitted successfully, LPTM is updated with its Mailbox number.
16 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
17 VPS	Valid Priority Status This bit indicates whether IMB and LPTM contents are currently valid or not. VPS is asserted upon every complete Tx arbitration process unless the CPU writes to Control and Status word of a Mailbox that has already been scanned, that is, it is behind Tx Arbitration Pointer, during the Tx arbitration process. If there is no inactive Mailbox and only one Tx Mailbox that is being transmitted then VPS is not asserted. VPS is negated upon the start of every Tx arbitration process or upon a write to Control and Status word of any Mailbox. NOTE: ESR2[VPS] is not affected by any CPU write into Control Status (C/S) of a MB that is blocked by abort mechanism. When MCR[AEN] is asserted, the abort code write in C/S of a MB that is being transmitted (pending abort), or any write attempt into a Tx MB with IFLAG set is blocked. 0 Contents of IMB and LPTM are invalid. 1 Contents of IMB and LPTM are valid.
18 IMB	Inactive Mailbox If ESR2[VPS] is asserted, this bit indicates whether there is any inactive Mailbox (CODE field is either 0b1000 or 0b0000). This bit is asserted in the following cases:

Table continues on the next page...

CAN_ESR2 field descriptions (continued)

Field	Description
	- During arbitration, if an LPTM is found and it is inactive. - If IMB is not asserted and a frame is transmitted successfully. This bit is cleared in all start of arbitration (see Section "Arbitration process"). NOTE: LPTM mechanism have the following behavior: if an MB is successfully transmitted and ESR2[IMB]=0 (no inactive Mailbox), then ESR2[VPS] and ESR2[IMB] are asserted and the index related to the MB just transmitted is loaded into ESR2[LPTM]. 0 If ESR2[VPS] is asserted, the ESR2[LPTM] is not an inactive Mailbox. 1 If ESR2[VPS] is asserted, there is at least one inactive Mailbox. LPTM content is the number of the first one.
19–31 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

40.4.16 CRC Register (CAN_CRCR)

This register provides information about the CRC of transmitted messages.

Address: 0h base + 44h offset = 44h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0								MBCRC							
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	TXCRC														
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

CAN_CRCR field descriptions

Field	Description
0–8 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
9–15 MBCRC	CRC Mailbox This field indicates the number of the Mailbox corresponding to the value in TXCRC field.
16 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
17–31 TXCRC	CRC Transmitted This field indicates the CRC value of the last message transmitted. This field is updated at the same time the Tx Interrupt Flag is asserted.

40.4.17 Rx FIFO Global Mask register (CAN_RXFGMASK)

This register is located in RAM.

If Rx FIFO is enabled RXFGMASK is used to mask the Rx FIFO ID Filter Table elements that do not have a corresponding RXIMR according to CTRL2[RFFN] field setting.

This register can only be written in Freeze mode as it is blocked by hardware in other modes.

Address: 0h base + 48h offset = 48h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	FGM[31:0]																															
W																																
Reset	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	

CAN_RXFGMASK field descriptions

Field	Description																																
0–31 FGM[31:0]	Rx FIFO Global Mask Bits These bits mask the ID Filter Table elements bits in a perfect alignment. The following table shows how the FGM bits correspond to each IDAF field. <table><tr><th rowspan="2">Rx FIFO ID Filter Table Elements Format (MCR[IDAM])</th><th colspan="6">Identifier Acceptance Filter Fields</th></tr><tr><th>RTR</th><th>IDE</th><th>RXIDA</th><th>RXIDB ¹</th><th>RXIDC ²</th><th>Reserved</th></tr><tr><td>A</td><td>FGM[31]</td><td>FGM[30]</td><td>FGM[29:1]</td><td>-</td><td>-</td><td>FGM[0]</td></tr><tr><td>B</td><td>FGM[31], FGM[15]</td><td>FGM[30], FGM[14]</td><td rowspan="2">-</td><td>FGM[29:16], FGM[13:0]</td><td>-</td><td rowspan="2">-</td></tr><tr><td>C</td><td>-</td><td>-</td><td>-</td><td>FGM[31:24], FGM[23:16], FGM[15:8], FGM[7:0]</td></tr></table> 1. If MCR[IDAM] field is equivalent to the format B only the fourteen most significant bits of the Identifier of the incoming frame are compared with the Rx FIFO filter. 2. If MCR[IDAM] field is equivalent to the format C only the eight most significant bits of the Identifier of the incoming frame are compared with the Rx FIFO filter. 0 The corresponding bit in the filter is "don't care." 1 The corresponding bit in the filter is checked.	Rx FIFO ID Filter Table Elements Format (MCR[IDAM])	Identifier Acceptance Filter Fields						RTR	IDE	RXIDA	RXIDB ¹	RXIDC ²	Reserved	A	FGM[31]	FGM[30]	FGM[29:1]	-	-	FGM[0]	B	FGM[31], FGM[15]	FGM[30], FGM[14]	-	FGM[29:16], FGM[13:0]	-	-	C	-	-	-	FGM[31:24], FGM[23:16], FGM[15:8], FGM[7:0]
Rx FIFO ID Filter Table Elements Format (MCR[IDAM])	Identifier Acceptance Filter Fields																																
	RTR	IDE	RXIDA	RXIDB ¹	RXIDC ²	Reserved																											
A	FGM[31]	FGM[30]	FGM[29:1]	-	-	FGM[0]																											
B	FGM[31], FGM[15]	FGM[30], FGM[14]	-	FGM[29:16], FGM[13:0]	-	-																											
C	-	-		-	FGM[31:24], FGM[23:16], FGM[15:8], FGM[7:0]																												

1. If MCR[IDAM] field is equivalent to the format B only the fourteen most significant bits of the Identifier of the incoming frame are compared with the Rx FIFO filter.
2. If MCR[IDAM] field is equivalent to the format C only the eight most significant bits of the Identifier of the incoming frame are compared with the Rx FIFO filter.

40.4.18 Rx FIFO Information Register (CAN_RXFIR)

RXFIR provides information on Rx FIFO.

This register is the port through which the CPU accesses the output of the RXFIR FIFO located in RAM. The RXFIR FIFO is written by the FlexCAN whenever a new message is moved into the Rx FIFO as well as its output is updated whenever the output of the Rx FIFO is updated with the next message. See Section "Rx FIFO" for instructions on reading this register.

Address: 0h base + 4Ch offset = 4Ch

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0																						IDHIT									
W																																
Reset	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*

* Notes:

- x = Undefined at reset.

CAN_RXFIR field descriptions

Field	Description
0–22 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
23–31 IDHIT	Identifier Acceptance Filter Hit Indicator This field indicates which Identifier Acceptance Filter was hit by the received message that is in the output of the Rx FIFO. If multiple filters match the incoming message ID then the first matching IDAF found (lowest number) by the matching process is indicated. This field is valid only while the IFLAG[BUF5I] is asserted.

40.4.19 Rx Individual Mask Registers (CAN_RXIMRn)

These registers are located in RAM.

RXIMR are used as acceptance masks for ID filtering in Rx MBs and the Rx FIFO. If the Rx FIFO is not enabled, one mask register is provided for each available Mailbox, providing ID masking capability on a per Mailbox basis.

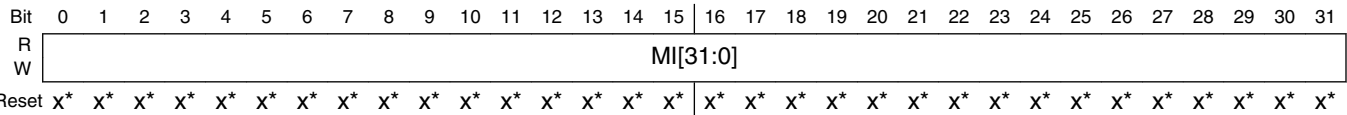
When the Rx FIFO is enabled (MCR[RFEN] bit is asserted), up to 32 Rx Individual Mask Registers can apply to the Rx FIFO ID Filter Table elements on a one-to-one correspondence depending on the setting of CTRL2[RFFN].

Memory map/register definition

RXIMR can only be written by the CPU while the module is in Freeze mode; otherwise, they are blocked by hardware.

The Individual Rx Mask Registers are not affected by reset and must be explicitly initialized prior to any reception.

Address: 0h base + 880h offset + (4d × i), where i=0d to 63d



- * Notes:
- x = Undefined at reset.

CAN_RXIMRn field descriptions

Field	Description
0–31 MI[31:0]	Individual Mask Bits Each Individual Mask Bit masks the corresponding bit in both the Mailbox filter and Rx FIFO ID Filter Table element in distinct ways. For Mailbox filters, see the RXMGMASK register description. For Rx FIFO ID Filter Table elements, see the RXFGMASK register description. 0 The corresponding bit in the filter is "don't care." 1 The corresponding bit in the filter is checked.

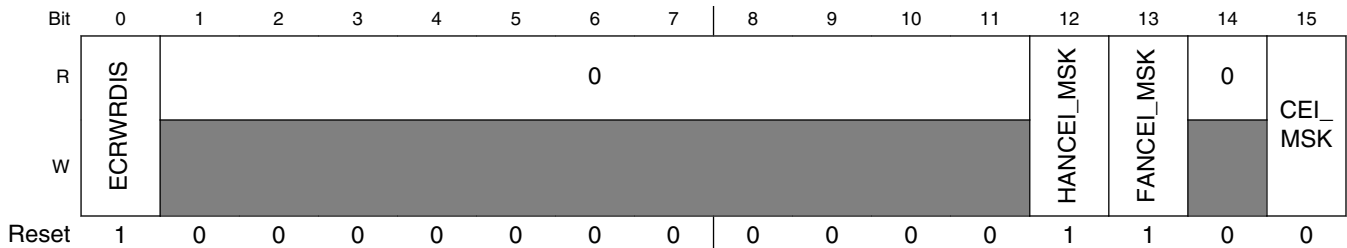
40.4.20 Memory Error Control Register (CAN_MECR)

This register contains control bits for memory error detection and correction (ECC).

NOTE

When bit CTRL2[ECRWRE] is 0, this register is read-only and writes to this register result in bus transfer errors.

Address: 0h base + AE0h offset = AE0h



Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	HAERRIE	FAERRIE	EXERRIE	0			RERRDIS	ECCDIS	NCEFAFRZ	0						
W																
Reset	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0

CAN_MECR field descriptions

Field	Description
0 ECRWRDIS	Error Configuration Register Write Disable Disables writes on this register. 0 Write is enabled. 1 Write is disabled.
1–11 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
12 HANCEI_MSK	Host Access With Non-Correctable Errors Interrupt Mask Enables the interrupt in case of non-correctable errors detected in memory reads issued by the host (CPU). 0 Interrupt is disabled. 1 Interrupt is enabled.
13 FANCEI_MSK	FlexCAN Access With Non-Correctable Errors Interrupt Mask Enables the interrupt in case of non-correctable errors detected in memory reads issued by the FlexCAN internal processes. 0 Interrupt is disabled. 1 Interrupt is enabled.
14 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
15 CEI_MSK	Correctable Errors Interrupt Mask Enables the interrupt in case of correctable errors detected in memory reads issued by the host or FlexCAN internal processes. 0 Interrupt is disabled. 1 Interrupt is enabled.
16 HAERRIE	Host Access Error Injection Enable Enables the injection of errors only in memory reads issued by the host (CPU). 0 Injection is disabled. 1 Injection is enabled.
17 FAERRIE	FlexCAN Access Error Injection Enable Enables the injection of errors only in memory reads issued by the FlexCAN internal processes. 0 Injection is disabled. 1 Injection is enabled.

Table continues on the next page...

CAN_MECR field descriptions (continued)

Field	Description
18 EXTERRIE	Extended Error Injection Enable Memory accesses performed by internal FlexCAN processes are 64-bit. This bit extends the error injection on 32-bit memory accesses to the complementary 32-bit word using the same 32-bit error injection data and parity words. See Error Injection Data Pattern Register (FCERRIDPR) and Error Injection Parity Pattern Register (FCERRIPPR) 0 Error injection is applied only to the 32-bit word. 1 Error injection is applied to the 64-bit word.
19–21 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
22 RERRDIS	Error Report Disable Disables the update of the error report registers. The update of error-related flags and the generation of bus transfer errors are still active. NOTE: When reading the report registers, this bit must be cleared to assure coherence on the consecutive register reads. 0 Enable updates of the error report registers. 1 Disable updates of the error report registers.
23 ECCDIS	Error Correction Disable Disables completely the memory error detection and correction mechanism. Besides disabling the error report mechanism, it also stops the update of the error-related flags and generation of bus transfer errors. The parity bits continue being calculated and written into memory on write transactions. 0 Enable memory error correction. 1 Disable memory error correction.
24 NCEFAFRZ	Non-Correctable Errors In FlexCAN Access Put Device In Freeze Mode Determines the response when a non-correctable error is detected in a memory read performed by FlexCAN internal processes. In this case, entering Freeze mode prevents corrupted data from being treated as valid by FlexCAN internal processes. 0 Keep normal operation. 1 Put FlexCAN in Freeze mode (according to the "Freeze mode" section).
25–31 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

40.4.21 Error Injection Address Register (CAN_ERRIAR)

This register holds the address where error is to be injected.

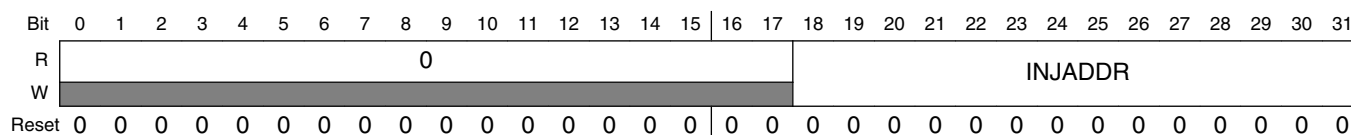
Use the following table to convert from the memory map address to the location in the physical FlexCAN RAM:

RAM contents	Injection address	Memory map
FlexCAN Registers	Not mapped	-

Table continues on the next page...

RAM contents	Injection address	Memory map
MBs	0x0000	0x0080
Reserved	-	0x0480
RXIMRs	0x0400	0x0880
Reserved	-	0x0980
RXFIR_0	0x0500	0x0A80
RXFIR_1	0x0504	0x0A84
RXFIR_2	0x0508	0x0A88
RXFIR_3	0x050C	0x0A8C
RXFIR_4	0x0510	0x0A90
RXFIR_5	0x0514	0x0A94
Reserved	-	0x0A98
RXMGMASK	0x0520	0x0AA0
RXFGMASK	0x0524	0x0AA4
RX14MASK	0x0528	0x0AA8
RX15MASK	0x052C	0x0AAC
SMB_TX	Not Mapped	0x0AB0
SMB_RX0	0x0540	0x0AC0
SMB_RX1	0x0550	0x0AD0
ECC Registers	Not mapped	0x0AE0
Reserved	-	0x0B00

Address: 0h base + AE4h offset = AE4h



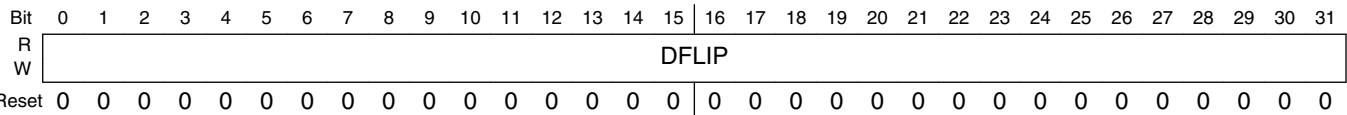
CAN_ERRIR field descriptions

Field	Description
0–17 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
18–31 INJADDR	Address Where Error Is To Be Injected The two least significant bits of this field always read as 0.

40.4.22 Error Injection Data Pattern Register (CAN_ERRIDPR)

Holds the error pattern to be injected in the data word read from memory.

Address: 0h base + AE8h offset = AE8h



CAN_ERRIDPR field descriptions

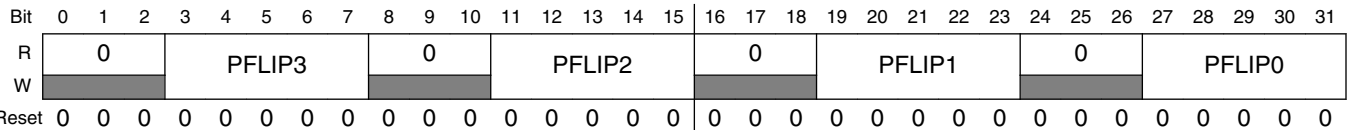
Field	Description
0–31 DFLIP	Data flip pattern Bits set to 1 in the flip pattern cause the corresponding data bit in the word read from memory to invert.

40.4.23 Error Injection Parity Pattern Register (CAN_ERRIPPR)

Holds the error pattern to be injected in parity bits read from memory along with data word.

Bits set to 1 in the flip pattern cause the corresponding parity bit in the word read from memory to invert.

Address: 0h base + AECh offset = AECh



CAN_ERRIPPR field descriptions

Field	Description
0–2 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
3–7 PFLIP3	Parity Flip Pattern For Byte 3 (most significant)
8–10 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
11–15 PFLIP2	Parity Flip Pattern For Byte 2
16–18 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

Table continues on the next page...

CAN_ERRIPPR field descriptions (continued)

Field	Description
19–23 PFLIP1	Parity Flip Pattern For Byte 1
24–26 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
27–31 PFLIP0	Parity Flip Pattern For Byte 0 (Least Significant)

40.4.24 Error Report Address Register (CAN_RERRAR)

Reports the address used for an access in which an error (correctable or non-correctable) was detected, and also reports the identification of the source of that access.

This address is always reported using a 32-bit alignment. Non-aligned accesses (ERRADDR[1:0] non-0) are reported with the address aligned and data is reported in RERRDR accordingly shifted. In case of errors detected in accesses larger than 32-bit (as performed by FlexCAN internal processes), the address of the 32-bit word in which the error was detected is reported. In case of errors detected in more than one 32-bit word, only the least significant address is reported.

Address: 0h base + AF0h offset = AF0h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	
R	0							NCE	0						SAID		
W																	
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0		ERRADDR													
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

CAN_RERRAR field descriptions

Field	Description
0–6 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
7 NCE	Non-Correctable Error Indicates that the report is due to an non-correctable error. 0 Reporting a correctable-error 1 Reporting a non-correctable error

Table continues on the next page...

CAN_RERRAR field descriptions (continued)

Field	Description														
8–12 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.														
13–15 SAID	SAID[2] — Identification of the requestor of the memory read request: 0 = Requested by FlexCAN internal processes 1 = Requested by Host (CPU) SAID[1] — Details of FlexCAN operation: 0 = Move 1 = Scanning SAID[0] — Operation that requested the memory read: 0 = Transmission 1 = Reception Table 40-91. Source of memory access <table> <tr> <th>SAID[2:0]</th><th>Error during...</th></tr> <tr> <td>0</td><td>Tx Move</td></tr> <tr> <td>1</td><td>Rx Move</td></tr> <tr> <td>2</td><td>Tx Arbitration</td></tr> <tr> <td>3</td><td>Rx Match</td></tr> <tr> <td>4</td><td>Tx Move</td></tr> <tr> <td>5-7</td><td>Reserved</td></tr> </table>	SAID[2:0]	Error during...	0	Tx Move	1	Rx Move	2	Tx Arbitration	3	Rx Match	4	Tx Move	5-7	Reserved
SAID[2:0]	Error during...														
0	Tx Move														
1	Rx Move														
2	Tx Arbitration														
3	Rx Match														
4	Tx Move														
5-7	Reserved														
16–17 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.														
18–31 ERRADDR	Address Where The Error Was Detected See the description of the Error Injection Address Register (ERRIAR).														

40.4.25 Error Report Data Register (CAN_RERRDR)

Reports the raw data (unmodified by the correction performed by ECC logic) read from memory with error. The value reported does not represent the transient values of the BUSY bit (see the “Message Buffer Code for Rx buffers” table) when reading a Message Buffer.

Address: 0h base + AF4h offset = AF4h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	RDATA																															
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	

CAN_RERRDR field descriptions

Field	Description
0–31 RDATA	Raw data word read from memory with error

40.4.26 Error Report Syndrome Register (CAN_RERRSYNR)

Holds the syndrome detected in a memory read with error. It also reports the bytes which were read in this 32-bit read transaction.

Each SYNDn field indicates the type of error and which bit in byte (n) is affected by the error. (SYND3 corresponds to the most significant byte in the data word read from memory; SYND0 corresponds to the least significant.)

Table 40-94. Syndrome Definition

SYNDn (hex)	Type	Bit affected
00	-	none (no error)
01	Code	0
02	Code	1
04	Code	2
07	Data	5
08	Code	3
0E	Data	7
10	Code	4
13	Data	2
15	Data	6
16	Data	1
19	Data	3
1A	Data	4
1C	Data	0
All others	-	Non-correctable error

Each BEn field indicates which byte in the 32-bit word reported was effectively read. The syndrome bits are calculated for all bytes, even for the non-read ones. Errors detected in non-read bytes are indicated (see the "Error Indication" section) and reported (see the "Error Reporting" section).

Memory map/register definition

Address: 0h base + AF8h offset = AF8h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	BE3	0							BE2	0						
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	BE1	0							BE0	0						
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

CAN_RERRSYNR field descriptions

Field	Description
0 BE3	Byte Enabled For Byte 3 (Most Significant) 0 The byte was not read. 1 The byte was read.
1–2 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
3–7 SYND3	Error Syndrome For Byte 3 (Most Significant) See the "Syndrome Definition" table.
8 BE2	Byte Enabled For Byte 2 0 The byte was not read. 1 The byte was read.
9–10 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
11–15 SYND2	Error Syndrome For Byte 2 See the "Syndrome Definition" table.
16 BE1	Byte Enabled For Byte 1 0 The byte was not read. 1 The byte was read.
17–18 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
19–23 SYND1	Error Syndrome for Byte 1 See the "Syndrome Definition" table.
24 BE0	Byte Enabled For Byte 0 (least significant) 0 The byte was not read. 1 The byte was read.
25–26 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
27–31 SYND0	Error Syndrome For Byte 0 (least significant) See the "Syndrome Definition" table.

40.4.27 Error Status Register (CAN_ERRSR)

Holds the status bits of the error correction and detection operations. After raised, these flags must be cleared by writing 1 to them. Writing 0 has no effect.

Address: 0h base + AFCh offset = AFCh

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0												HANCEIF	FANCEIF	0	CEIF
W													w1c	w1c		w1c
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0												HANCEIOF	FANCEIOF	0	CEIOF
W													w1c	w1c		w1c
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

CAN_ERRSR field descriptions

Field	Description
0–11 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
12 HANCEIF	Host Access With Non-Correctable Error Interrupt Flag Indicates that a non-correctable error was detected in a memory access initiated by Host. A bus transfer error is asserted for that access. If MECR[HANCEI_MSK] is set, the interrupt is asserted. 0 No non-correctable errors were detected in Host accesses so far. 1 A non-correctable error was detected in a Host access.
13 FANCEIF	FlexCAN Access With Non-Correctable Error Interrupt Flag

Table continues on the next page...

CAN_ERRSR field descriptions (continued)

Field	Description
	Indicates that a non-correctable error was detected in a memory access initiated by FlexCAN internal processes. If MECR[FANCEI_MSK] is set, the interrupt is asserted. 0 No non-correctable errors were detected in FlexCAN accesses so far. 1 A non-correctable error was detected in a FlexCAN access.
14 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
15 CEIF	Correctable Error Interrupt Flag Indicates that a correctable error was detected in a memory access. If MECR[CEI_MSK] is set, the interrupt is asserted. 0 No correctable errors were detected so far. 1 A correctable error was detected.
16–27 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
28 HANCEIOF	Host Access With Non-Correctable Error Interrupt Overrun Flag Indicates that a non-correctable error was detected in a memory access initiated by Host when HANCEIF was set. No interrupt is associated to this flag. (See the "Error Indication" section.) 0 No overrun on non-correctable errors in Host access 1 Overrun on non-correctable errors in Host access
29 FANCEIOF	FlexCAN Access With Non-Correctable Error Interrupt Overrun Flag Indicates that a non-correctable error was detected in a memory access initiated by FlexCAN internal processes when FANCEIF was set. No interrupt is associated to this flag. (See the "Error Indication" section.) 0 No overrun on non-correctable errors in FlexCAN access 1 Overrun on non-correctable errors in FlexCAN access
30 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
31 CEIOF	Correctable Error Interrupt Overrun Flag Indicates that a correctable error was detected in a memory access when CEIF was set. No interrupt is associated to this flag. (See the "Error Indication" section.) 0 No overrun on correctable errors 1 Overrun on correctable errors

40.4.28 Message buffer structure

The message buffer structure used by the FlexCAN module is represented in the following figure. Both Extended (29-bit identifier) and Standard (11-bit identifier) frames used in the CAN specification (Version 2.0 Part B) are represented. Each individual MB is formed by 16 bytes.

The memory area from 0x80 to 0x47F is used by the mailboxes.

Table 40-96. Message buffer structure

	0	1	2	3	4	7	8	9	10	11	12	13	14	15	16	23	24	31	
0x0					CODE			SRR	IDE	RTR	DLC				TIME STAMP				
0x4	PRIO			ID (Standard/Extended)										ID (Extended)					
0x8	Data Byte 0						Data Byte 1						Data Byte 2			Data Byte 3			
0xC	Data Byte 4						Data Byte 5						Data Byte 6			Data Byte 7			
					= Unimplemented or Reserved														

CODE — Message Buffer Code

This 4-bit field can be accessed (read or write) by the CPU and by the FlexCAN module itself, as part of the message buffer matching and arbitration process. The encoding is shown in [Table 40-97](#) and [Table 40-98](#). See [Functional description](#) for additional information.

Table 40-97. Message buffer code for Rx buffers

CODE description	Rx code BEFORE receive new frame	SRV ¹	Rx code AFTER successful reception ²	RRS ³	Comment
0b0000: INACTIVE — MB is not active.	INACTIVE	—	—	—	MB does not participate in the matching process.
0b0100: EMPTY — MB is active and empty.	EMPTY	—	FULL	—	When a frame is received successfully (after the Move-in process), the CODE field is automatically updated to FULL.

Table continues on the next page...

Table 40-97. Message buffer code for Rx buffers (continued)

CODE description	Rx code BEFORE receive new frame	SRV ¹	Rx code AFTER successful reception ²	RRS ³	Comment
0b0010: FULL — MB is full.	FULL	Yes	FULL	—	The act of reading the C/S word followed by unlocking the MB (SRV) does not make the code return to EMPTY. It remains FULL. If a new frame is moved to the MB after the MB was serviced, the code still remains FULL. See Matching process for matching details related to FULL code.
		No	OVERRUN	—	If the MB is FULL and a new frame is moved to this MB before the CPU services it, the CODE field is automatically updated to OVERRUN. See Matching process for details about overrun behavior.
0b0110: OVERRUN — MB is being overwritten into a full buffer.	OVERRUN	Yes	FULL	—	If the CODE field indicates OVERRUN and CPU has serviced the MB, when a new frame is moved to the MB then the code returns to FULL.
		No	OVERRUN	—	If the CODE field already indicates OVERRUN, and another new frame must be moved, the MB will be overwritten again, and the code will remain OVERRUN. See Matching process for details about overrun behavior.

Table continues on the next page...

Table 40-97. Message buffer code for Rx buffers (continued)

CODE description	Rx code BEFORE receive new frame	SRV ¹	Rx code AFTER successful reception ²	RRS ³	Comment
0b1010: RANSWER ⁴ — A frame was configured to recognize a Remote Request Frame and transmit a Response Frame in return.	RANSWER	—	TANSWER(0b1110)	0	A Remote Answer was configured to recognize a remote request frame received. After that an MB is set to transmit a response frame. The code is automatically changed to TANSWER (0b1110). See Matching process for details. If CTRL2[RRS] is negated, transmit a response frame whenever a remote request frame with the same ID is received.
		—	—	1	This code is ignored during matching and arbitration process. See Matching process for details.
CODE[0]=1b1: BUSY ⁵ — FlexCAN is updating the contents of the MB. The CPU must not access the MB.	BUSY ⁵	—	FULL	—	Indicates that the MB is being updated. It will be negated automatically and does not interfere with the next CODE.
		—	OVERRUN	—	

1. SRV: Serviced MB. MB was read and unlocked by reading TIMER or other MB.
2. A frame is considered a successful reception after the frame to be moved to MB (move-in process). See [Move-in](#) for details.
3. Remote Request Stored bit from CTRL2 register. See section "Control 2 Register (CTRL2)" for details.
4. Code 0b1010 is not considered Tx and an MB with this code should not be aborted.
5. Note that for Tx MBs, the BUSY bit should be ignored upon read, except when AEN bit is set in the MCR register. If this bit is asserted, the corresponding MB does not participate in the matching process.

Table 40-98. Message buffer code for Tx buffers

CODE Description	Tx Code BEFORE tx frame	MB RTR	Tx Code AFTER successful transmission	Comment
0b1000: INACTIVE — MB is not active	INACTIVE	—	—	MB does not participate in arbitration process.

Table continues on the next page...

Table 40-98. Message buffer code for Tx buffers (continued)

CODE Description	Tx Code BEFORE tx frame	MB RTR	Tx Code AFTER successful transmission	Comment
0b1001: ABORT — MB is aborted	ABORT	—	—	MB does not participate in arbitration process.
0b1100: DATA — MB is a Tx Data Frame (MB RTR must be 0)	DATA	0	INACTIVE	Transmit data frame unconditionally once. After transmission, the MB automatically returns to the INACTIVE state.
0b1100: REMOTE — MB is a Tx Remote Request Frame (MB RTR must be 1)	REMOTE	1	EMPTY	Transmit remote request frame unconditionally once. After transmission, the MB automatically becomes an Rx Empty MB with the same ID.
0b1110: TANSWER — MB is a Tx Response Frame from an incoming Remote Request Frame	TANSWER	—	RANSWER	This is an intermediate code that is automatically written to the MB by the CHI as a result of a match to a remote request frame. The remote response frame will be transmitted unconditionally once, and then the code will automatically return to RANSWER (0b1010). The CPU can also write this code with the same effect. The remote response frame can be either a data frame or another remote request frame depending on the RTR bit value. See Matching process and Arbitration process for details.

SRR — Substitute Remote Request

Fixed recessive bit, used only in extended format. It must be set to one by the user for transmission (Tx Buffers) and will be stored with the value received on the CAN bus for Rx receiving buffers. It can be received as either recessive or dominant. If FlexCAN receives this bit as dominant, then it is interpreted as an arbitration loss.

1 = Recessive value is compulsory for transmission in extended format frames

0 = Dominant is not a valid value for transmission in extended format frames

IDE — ID Extended Bit

This field identifies whether the frame format is standard or extended.

1 = Frame format is extended

0 = Frame format is standard

RTR — Remote Transmission Request

This bit affects the behavior of remote frames and is part of the reception filter. See [Table 40-97](#), [Table 40-98](#), and the description of the RRS bit in Control 2 Register (CTRL2) for additional details.

If FlexCAN transmits this bit as '1' (recessive) and receives it as '0' (dominant), it is interpreted as an arbitration loss. If this bit is transmitted as '0' (dominant), then if it is received as '1' (recessive), the FlexCAN module treats it as a bit error. If the value received matches the value transmitted, it is considered a successful bit transmission.

1 = Indicates the current MB may have a remote request frame to be transmitted if MB is Tx. If the MB is Rx then incoming remote request frames may be stored.

0 = Indicates the current MB has a data frame to be transmitted. In Rx MB it may be considered in matching processes.

DLC — Length of Data in Bytes

This 4-bit field is the length (in bytes) of the Rx or Tx data, which is located in offset 0x8 through 0xF of the MB space (see the [Table 40-96](#)). In reception, this field is written by the FlexCAN module, copied from the DLC (Data Length Code) field of the received frame. In transmission, this field is written by the CPU and corresponds to the DLC field value of the frame to be transmitted. When RTR = 1, the frame to be transmitted is a remote frame and does not include the data field, regardless of the DLC field (see [Table 40-99](#)).

TIME STAMP — Free-Running Counter Time Stamp

This 16-bit field is a copy of the Free-Running Timer, captured for Tx and Rx frames at the time when the beginning of the Identifier field appears on the CAN bus.

PRI0 — Local priority

This 3-bit field is used only when LPRIO_EN bit is set in MCR, and it only makes sense for Tx mailboxes. These bits are not transmitted. They are appended to the regular ID to define the transmission priority. See [Arbitration process](#).

ID — Frame Identifier

In standard frame format, only the 11 most significant bits (3 to 13) are used for frame identification in both receive and transmit cases. The 18 least significant bits are ignored. In extended frame format, all bits are used for frame identification in both receive and transmit cases.

DATA BYTE 0–7 — Data Field

Up to eight bytes can be used for a data frame.

For Rx frames, the data is stored as it is received from the CAN bus. DATA BYTE (n) is valid only if n is less than DLC as shown in the table below.

For Tx frames, the CPU prepares the data field to be transmitted within the frame.

Table 40-99. DATA BYTEs validity

DLC	Valid DATA BYTEs
0	none
1	DATA BYTE 0
2	DATA BYTE 0–1
3	DATA BYTE 0–2
4	DATA BYTE 0–3
5	DATA BYTE 0–4
6	DATA BYTE 0–5
7	DATA BYTE 0–6
8	DATA BYTE 0–7

40.4.29 Rx FIFO structure

When the MCR[RFEN] bit is set, the memory area from 0x80 to 0xDC (which is normally occupied by MBs 0–5) is used by the reception FIFO engine.

The region 0x80-0x8C contains the output of the FIFO which must be read by the CPU as a message buffer. This output contains the oldest message that has been received but not yet read. The region 0x90-0xDC is reserved for internal use of the FIFO engine.

An additional memory area, which starts at 0xE0 and may extend up to 0x2DC (normally occupied by MBs 6–37) depending on the CTRL2[RFFN] field setting, contains the ID filter table (configurable from 8 to 128 table elements) that specifies filtering criteria for accepting frames into the FIFO.

Out of reset, the ID filter table flexible memory area defaults to 0xE0 and extends only to 0xFC, which corresponds to MBs 6 to 7 for RFFN = 0, for backward compatibility with previous versions of FlexCAN.

The following shows the Rx FIFO data structure.

Table 40-100. Rx FIFO structure

	0	3	7	8	9	10	11	12	13	14	15	16	23	24	31			
0x80					SRR	IDE	RTR	DLC				TIME STAMP						
0x84		ID standard								ID extended								
0x88	Data byte 0				Data byte 1								Data byte 2				Data byte 3	
0x8C	Data byte 4				Data byte 5								Data byte 6				Data byte 7	
0x90 to 0xDC	Reserved																	
0xE0	ID filter table element 0																	
0xE4	ID filter table element 1																	
0xE8 to 0x2D4	ID filter table elements 2 to 125																	
0x2D8	ID filter table element 126																	
0x2DC	ID filter table element 127																	
		= Unimplemented or reserved																

Each ID filter table element occupies an entire 32-bit word and can be compounded by one, two, or four Identifier Acceptance Filters (IDAF) depending on the MCR[IDAM] field setting. The following figures show the IDAF indexation.

The following table shows the three different formats of the ID table elements. Note that all elements of the table must have the same format. See [Rx FIFO](#) for more information.

Table 40-101. ID table structure

Format	0	1	2	7	8	15	16	17	18	23	24	30	31
A	RTR	IDE	RXIDA (standard = 2–12, extended = 2–30)										
B	RTR	IDE	RXIDB_0 (standard = 2–12, extended = 2–15)				RTR	IDE	RXIDB_1 (standard = 18–28, extended = 18–31)				
C	RXIDC_0 (std/ext = 0–7)				RXIDC_1 (std/ext = 8–15)			RXIDC_2 (std/ext = 16–23)				RXIDC_3 (std/ext = 24–31)	
	= Unimplemented or Reserved												

RTR — Remote Frame

This bit specifies if Remote Frames are accepted into the FIFO if they match the target ID.

1 = Remote Frames can be accepted and data frames are rejected

0 = Remote Frames are rejected and data frames can be accepted

IDE — Extended Frame

Specifies whether extended or standard frames are accepted into the FIFO if they match the target ID.

1 = Extended frames can be accepted and standard frames are rejected

0 = Extended frames are rejected and standard frames can be accepted

RXIDA — Rx Frame Identifier (Format A)

Specifies an ID to be used as acceptance criteria for the FIFO. In the standard frame format, only the 11 most significant bits (2 to 12) are used for frame identification. In the extended frame format, all bits are used.

RXIDB_0, RXIDB_1 — Rx Frame Identifier (Format B)

Specifies an ID to be used as acceptance criteria for the FIFO. In the standard frame format, the 11 most significant bits (a full standard ID) (2 to 12 and 18 to 28) are used for frame identification. In the extended frame format, all 14 bits of the field are compared to the 14 most significant bits of the received ID.

RXIDC_0, RXIDC_1, RXIDC_2, RXIDC_3 — Rx Frame Identifier (Format C)

Specifies an ID to be used as acceptance criteria for the FIFO. In both standard and extended frame formats, all 8 bits of the field are compared to the 8 most significant bits of the received ID.

40.5 Functional description

The FlexCAN module is a CAN protocol engine with a very flexible mailbox system for transmitting and receiving CAN frames. The mailbox system is composed by a set of Message Buffers (MB) that store configuration and control data, time stamp, message ID and data (see [Message buffer structure](#)). The memory corresponding to the first 38 MBs can be configured to support a FIFO reception scheme with a powerful ID filtering mechanism, capable of checking incoming frames against a table of IDs (up to 128 extended IDs or 256 standard IDs or 512 8-bit ID slices), with individual mask register for up to 32 ID Filter Table elements. Simultaneous reception through FIFO and mailbox is supported. For mailbox reception, a matching algorithm makes it possible to store received frames only into MBs that have the same ID programmed on its ID field. A masking scheme makes it possible to match the ID programmed on the MB with a range

of IDs on received CAN frames. For transmission, an arbitration algorithm decides the prioritization of MBs to be transmitted based on the message ID (optionally augmented by 3 local priority bits) or the MB ordering.

Before proceeding with the functional description, an important concept must be explained. A Message Buffer is said to be "active" at a given time if it can participate in both the Matching and Arbitration processes. An Rx MB with a 0b0000 code is inactive (refer to [Table 40-97](#)). Similarly, a Tx MB with a 0b1000 or 0b1001 code is also inactive (refer to [Table 40-98](#)).

40.5.1 Transmit process

To transmit a CAN frame, the CPU must prepare a Message Buffer for transmission by executing the following procedure:

1. Check whether the respective interrupt bit is set and clear it.
2. If the MB is active (transmission pending), write the ABORT code (0b1001) to the CODE field of the Control and Status word to request an abortion of the transmission. Wait for the corresponding IFLAG to be asserted by polling the IFLAG register or by the interrupt request if enabled by the respective IMASK. Then read back the CODE field to check if the transmission was aborted or transmitted (see [Transmission abort mechanism](#)). If backwards compatibility is desired (MCR[AEN] bit is negated), just write the INACTIVE code (0b1000) to the CODE field to inactivate the MB but then the pending frame may be transmitted without notification (see [Mailbox inactivation](#)).
3. Write the ID word.
4. Write the data bytes.
5. Write the DLC, Control, and CODE fields of the Control and Status word to activate the MB.

When the MB is activated, it will participate into the arbitration process and eventually be transmitted according to its priority. At the end of the successful transmission, the value of the Free Running Timer is written into the Time Stamp field, the CODE field in the Control and Status word is updated, the CRC Register is updated, a status flag is set in the Interrupt Flag Register and an interrupt is generated if allowed by the corresponding Interrupt Mask Register bit. The new CODE field after transmission depends on the code that was used to activate the MB (see [Table 40-97](#) and [Table 40-98](#) in [Message buffer structure](#)).

When the Abort feature is enabled (MCR[AEN] is asserted), after the Interrupt Flag is asserted for a Mailbox configured as transmit buffer, the Mailbox is blocked, therefore the CPU is not able to update it until the Interrupt Flag is negated by CPU. This means that the CPU must clear the corresponding IFLAG before starting to prepare this MB for a new transmission or reception.

40.5.2 Arbitration process

The arbitration process scans the Mailboxes searching the Tx one that holds the message to be sent in the next opportunity. This Mailbox is called the *arbitration winner*.

The scan starts from the lowest number Mailbox and runs toward the higher ones.

The arbitration process is triggered in the following events:

- From the CRC field of the CAN frame. The start point depends on the CTRL2[TASD] field value.
- During the Error Delimiter field of a CAN frame.
- During the Overload Delimiter field of a CAN frame.
- When the winner is inactivated and the CAN bus has still not reached the first bit of the Intermission field.
- When there is CPU write to the C/S word of a winner MB and the CAN bus has still not reached the first bit of the Intermission field.
- When CHI is in Idle state and the CPU writes to the C/S word of any MB.
- When FlexCAN exits Bus Off state.
- Upon leaving Freeze mode or Low Power mode.

If the arbitration process does not manage to evaluate all Mailboxes before the CAN bus has reached the first bit of the Intermission field the temporary arbitration winner is invalidated and the FlexCAN will not compete for the CAN bus in the next opportunity.

The arbitration process selects the winner among the active Tx Mailboxes at the end of the scan according to both CTRL1[LBUF] and MCR[LPRIOEN] bits settings.

40.5.2.1 Lowest-number Mailbox first

If CTRL1[LBUF] bit is asserted the first (lowest number) active Tx Mailbox found is the arbitration winner. MCR[LPRIOEN] bit has no effect when CTRL1[LBUF] is asserted.

40.5.2.2 Highest-priority Mailbox first

If CTRL1[LBUF] bit is negated, then the arbitration process searches the active Tx Mailbox with the highest priority, which means that this Mailbox's frame would have a higher probability to win the arbitration on CAN bus when multiple external nodes compete for the bus at the same time.

The sequence of bits considered for this arbitration is called the *arbitration value* of the Mailbox. The highest-priority Tx Mailbox is the one that has the lowest arbitration value among all Tx Mailboxes.

If two or more Mailboxes have equivalent arbitration values, the Mailbox with the lowest number is the arbitration winner.

The composition of the arbitration value depends on MCR[LPRIOEN] bit setting.

40.5.2.2.1 Local Priority disabled

If MCR[LPRIOEN] bit is negated the arbitration value is built in the exact sequence of bits as they would be transmitted in a CAN frame (see the following table) in such a way that the Local Priority is disabled.

Table 40-102. Composition of the arbitration value when Local Priority is disabled

Format	Mailbox Arbitration Value (32 bits)				
Standard (IDE = 0)	Standard ID (11 bits)	RTR (1 bit)	IDE (1 bit)	- (18 bits)	- (1 bit)
Extended (IDE = 1)	Extended ID[28:18] (11 bits)	SRR (1 bit)	IDE (1 bit)	Extended ID[17:0] (18 bits)	RTR (1 bit)

40.5.2.2.2 Local Priority enabled

If Local Priority is desired MCR[LPRIOEN] must be asserted. In this case the Mailbox PRIO field is included at the very left of the arbitration value (see the following table).

Table 40-103. Composition of the arbitration value when Local Priority is enabled

Format	Mailbox Arbitration Value (35 bits)					
Standard (IDE = 0)	PRI0 (3 bits)	Standard ID (11 bits)	RTR (1 bit)	IDE (1 bit)	- (18 bits)	- (1 bit)
Extended (IDE = 1)	PRI0 (3 bits)	Extended ID[28:18] (11 bits)	SRR (1 bit)	IDE (1 bit)	Extended ID[17:0] (18 bits)	RTR (1 bit)

As the PRI0 field is the most significant part of the arbitration value Mailboxes with low PRI0 values have higher priority than Mailboxes with high PRI0 values regardless the rest of their arbitration values.

Note that the PRI0 field is not part of the frame on the CAN bus. Its purpose is only to affect the internal arbitration process.

40.5.2.3 Arbitration process (continued)

After the arbitration winner is found, its content is copied to a hidden auxiliary MB called Tx Serial Message Buffer (Tx SMB), which has the same structure as a normal MB but is not user accessible. This operation is called move-out and after it is done, write access to the corresponding MB is blocked (if the AEN bit in MCR is asserted). The write access is released in the following events:

- After the MB is transmitted
- FlexCAN enters in Freeze mode or Bus Off
- FlexCAN loses the bus arbitration or there is an error during the transmission

At the first opportunity window on the CAN bus, the message on the Tx SMB is transmitted according to the CAN protocol rules. FlexCAN transmits up to eight data bytes, even if the Data Length Code (DLC) field value is greater than that.

Arbitration process can be triggered in the following situations:

- During Rx and Tx frames from CAN CRC field to end of frame. T ASD value may be changed to optimize the arbitration start point.
- During CAN BusOff state from TX_ERR_CNT=124 to 128. T ASD value may be changed to optimize the arbitration start point.
- During C/S write by CPU in BusIdle. First C/S write starts arbitration process and a second C/S write during this same arbitration restarts the process. If other C/S writes are performed, Tx arbitration process is pending. If there is no arbitration winner after arbitration process has finished, then TX arbitration machine begins a new arbitration process.

- If there is a pending arbitration and BusIdle state starts then an arbitration process is triggered. In this case the first and second C/S write in BusIdle will not restart the arbitration process. It is possible that there is not enough time to finish arbitration in WaitForBusIdle state and the next state is Idle. In this case the scan is not interrupted, and it is completed during BusIdle state. During this arbitration C/S write does not cause arbitration restart.
- Arbitration winner deactivation during a valid arbitration window.
- Upon Leave Freeze mode (first bit of the WaitForBusIdle state). If there is a re-synchronization during WaitForBusIdle arbitration process is restarted.

Arbitration process stops in the following situation:

- All Mailboxes were scanned
- A Tx active Mailbox is found in case of Lowest Buffer feature enabled
- Arbitration winner inactivation or abort during any arbitration process
- There was not enough time to finish Tx arbitration process (for instance, when a deactivation was performed near the end of frame). In this case arbitration process is pending.
- Error or Overload flag in the bus
- Low Power or Freeze mode request in Idle state

Arbitration is considered pending as described below:

- It was not possible to finish arbitration process in time
- C/S write during arbitration if write is performed in a MB whose number is lower than the Tx arbitration pointer
- Any C/S write if there is no Tx Arbitration process in progress
- Rx Match has just updated a Rx Code to Tx Code
- Entering Busoff state

C/S write during arbitration has the following effect:

- If C/S write is performed in the arbitration winner, a new process is restarted immediately.
- If C/S write is performed in a MB whose number is higher than the Tx arbitration pointer, the ongoing arbitration process will scan this MB as normal.

40.5.3 Receive process

To be able to receive CAN frames into a Mailbox, the CPU must prepare it for reception by executing the following steps:

1. If the Mailbox is active (either Tx or Rx) inactivate the Mailbox (see [Mailbox inactivation](#)), preferably with a safe inactivation (see [Transmission abort mechanism](#)).
2. Write the ID word
3. Write the EMPTY code (0b0100) to the CODE field of the Control and Status word to activate the Mailbox.

After the MB is activated, it will be able to receive frames that match the programmed filter. At the end of a successful reception, the Mailbox is updated by the *move-in* process (see [Move-in](#)) as follows:

1. The received Data field (8 bytes at most) is stored.
2. The received Identifier field is stored.
3. The value of the Free Running Timer at the time of the second bit of frame's Identifier field is written into the Mailbox's Time Stamp field.
4. The received SRR, IDE, RTR, and DLC fields are stored.
5. The CODE field in the Control and Status word is updated (see [Table 40-97](#) and [Table 40-98](#) in Section [Message buffer structure](#)).
6. A status flag is set in the Interrupt Flag Register and an interrupt is generated if allowed by the corresponding Interrupt Mask Register bit.

The recommended way for CPU servicing (read) the frame received in an Mailbox is using the following procedure:

1. Read the Control and Status word of that Mailbox.
2. Check if the BUSY bit is deasserted, indicating that the Mailbox is locked. Repeat step 1) while it is asserted. See [Mailbox lock mechanism](#).
3. Read the contents of the Mailbox. Once Mailbox is locked now, its contents won't be modified by FlexCAN Move-in processes. See [Move-in](#).
4. Acknowledge the proper flag at IFLAG registers.
5. Read the Free Running Timer. It is optional but recommended to unlock Mailbox as soon as possible and make it available for reception.

The CPU should synchronize to frame reception by the status flag bit for the specific Mailbox in one of the IFLAG Registers and not by the CODE field of that Mailbox. Polling the CODE field does not work because once a frame was received and the CPU

services the Mailbox (by reading the C/S word followed by unlocking the Mailbox), the CODE field will not return to EMPTY. It will remain FULL, as explained in [Table 40-97](#). If the CPU tries to workaround this behavior by writing to the C/S word to force an EMPTY code after reading the Mailbox without a prior *safe inactivation*, a newly received frame matching the filter of that Mailbox may be lost.

CAUTION

In summary: never do polling by reading directly the C/S word of the Mailboxes. Instead, read the IFLAG registers.

Note that the received frame's Identifier field is always stored in the matching Mailbox, thus the contents of the ID field in an Mailbox may change if the match was due to masking. Note also that FlexCAN does receive frames transmitted by itself if there exists a matching Rx Mailbox, provided the MCR[SRXDIS] bit is not asserted. If the MCR[SRXDIS] bit is asserted, FlexCAN will not store frames transmitted by itself in any MB, even if it contains a matching MB, and no interrupt flag or interrupt signal will be generated due to the frame reception.

To be able to receive CAN frames through the Rx FIFO, the CPU must enable and configure the Rx FIFO during Freeze mode (see [Rx FIFO](#)). Upon receiving the Frames Available in Rx FIFO interrupt (see the description of the IFLAG[BUF5I] "Frames available in Rx FIFO" bit in the IMASK1 register), the CPU should service the received frame using the following procedure:

1. Read the Control and Status word (optional – needed only if a mask was used for IDE and RTR bits)
2. Read the ID field (optional – needed only if a mask was used)
3. Read the Data field
4. Read the RXFIR register (optional)
5. Clear the Frames Available in Rx FIFO interrupt by writing 1 to IFLAG[BUF5I] bit (mandatory – releases the MB and allows the CPU to read the next Rx FIFO entry)

40.5.4 Matching process

The matching process scans the MB memory looking for Rx MBs programmed with the same ID as the one received from the CAN bus. If the FIFO is enabled, the priority of scanning can be selected between Mailboxes and FIFO filters. In any case, the matching starts from the lowest number Message Buffer toward the higher ones. If no match is

found within the first structure then the other is scanned subsequently. In the event that the FIFO is full, the matching algorithm will always look for a matching MB outside the FIFO region.

As the frame is being received, it is stored in a hidden auxiliary MB called Rx Serial Message Buffer (Rx SMB).

The matching process start point depends on the following conditions:

- If the received frame is a remote frame, the start point is the CRC field of the frame
- If the received frame is a data frame with DLC field equal to zero, the start point is the CRC field of the frame
- If the received frame is a data frame with DLC field different than zero, the start point is the DATA field of the frame

If a matching ID is found in the FIFO table or in one of the Mailboxes, the contents of the SMB will be transferred to the FIFO or to the matched Mailbox by the move-in process. If any CAN protocol error is detected then no match results will be transferred to the FIFO or to the matched Mailbox at the end of reception.

The matching process scans all matching elements of both Rx FIFO (if enabled) and active Rx Mailboxes (CODE is EMPTY, FULL, OVERRUN or RANSWER) in search of a successful comparison with the matching elements of the Rx SMB that is receiving the frame on the CAN bus. The SMB has the same structure of a Mailbox. The reception structures (Rx FIFO or Mailboxes) associated with the matching elements that had a successful comparison are the *matched structures*. The *matching winner* is selected at the end of the scan among those matched structures and depends on conditions described ahead. See the following table.

Table 40-104. Matching architecture

Structure	SMB[RTR]	CTRL2[RRS]	CTRL2[EAC EN]	MB[IDE]	MB[RTR]	MB[ID ¹]	MB[CODE]
Mailbox	0	-	0	cmp ²	no_cmp ³	cmp_msk ⁴	EMPTY or FULL or OVERRUN
Mailbox	0	-	1	cmp_msk	cmp_msk	cmp_msk	EMPTY or FULL or OVERRUN
Mailbox	1	0	-	cmp	no_cmp	cmp	RANSWER
Mailbox	1	1	0	cmp	no_cmp	cmp_msk	EMPTY or FULL or OVERRUN

Table continues on the next page...

Table 40-104. Matching architecture (continued)

Structure	SMB[RTR]	CTRL2[RRS]	CTRL2[EAC EN]	MB[IDE]	MB[RTR]	MB[ID ¹]	MB[CODE]
Mailbox	1	1	1	cmp_msk	cmp_msk	cmp_msk	EMPTY or FULL or OVERRUN
FIFO ⁵	-	-	-	cmp_msk	cmp_msk	cmp_msk	-

1. For Mailbox structure, If SMB[IDE] is asserted, the ID is 29 bits (ID Standard + ID Extended). If SMB[IDE] is negated, the ID is only 11 bits (ID Standard). For FIFO structure, the ID depends on IDAM.
2. cmp: Compares the SMB contents with the MB contents regardless the masks.
3. no_cmp: The SMB contents are not compared with the MB contents
4. cmp_msk: Compares the SMB contents with MB contents taking into account the masks.
5. SMB[IDE] and SMB[RTR] are not taken into account when IDAM is type C.

A reception structure is *free-to-receive* when any of the following conditions is satisfied:

- The CODE field of the Mailbox is EMPTY
- The CODE field of the Mailbox is either FULL or OVERRUN and it has already been serviced (the C/S word was read by the CPU and unlocked as described in [Mailbox lock mechanism](#))
- The CODE field of the Mailbox is either FULL or OVERRUN and an inactivation (see [Mailbox inactivation](#)) is performed
- The Rx FIFO is not full

The scan order for Mailboxes and Rx FIFO is from the matching element with lowest number to the higher ones.

The matching winner search for Mailboxes is affected by the MCR[IRMQ] bit. If it is negated the matching winner is the first matched Mailbox regardless if it is free-to-receive or not. If it is asserted, the matching winner is selected according to the priority below:

1. the first free-to-receive matched Mailbox;
2. the last non free-to-receive matched Mailbox.

It is possible to select the priority of scan between Mailboxes and Rx FIFO by the CTRL2[MRP] bit.

If the selected priority is Rx FIFO first:

- If the Rx FIFO is a matched structure and is free-to-receive then the Rx FIFO is the matching winner regardless of the scan for Mailboxes
- Otherwise (the Rx FIFO is not a matched structure or is not free-to-receive), then the matching winner is searched among Mailboxes as described above

If the selected priority is Mailboxes first:

Functional description

- If a free-to-receive matched Mailbox is found, it is the matching winner regardless the scan for Rx FIFO
- If no matched Mailbox is found, then the matching winner is searched in the scan for the Rx FIFO
- If both conditions above are not satisfied and a non free-to-receive matched Mailbox is found then the matching winner determination is conditioned by the MCR[IRMQ] bit:
 - If MCR[IRMQ] bit is negated the matching winner is the first matched Mailbox
 - If MCR[IRMQ] bit is asserted the matching winner is the Rx FIFO if it is a free-to-receive matched structure, otherwise the matching winner is the last non free-to-receive matched Mailbox

See the following table for a summary of matching possibilities.

Table 40-105. Matching possibilities and resulting reception structures

RFEN	IRMQ	MRP	Matched in MB	Matched in FIFO	Reception structure	Description
No FIFO, only MB, match is always MB first						
0	0	X ¹	None ²	- ³	None	Frame lost by no match
0	0	X	Free ⁴	-	FirstMB	
0	1	X	None	-	None	Frame lost by no match
0	1	X	Free	-	FirstMb	
0	1	X	NotFree	-	LastMB	Overrun
FIFO enabled, no match in FIFO is as if FIFO does not exist						
1	0	X	None	None ⁵	None	Frame lost by no match
1	0	X	Free	None	FirstMB	
1	1	X	None	None	None	Frame lost by no match
1	1	X	Free	None	FirstMb	
1	1	X	NotFree	None	LastMB	Overrun
FIFO enabled, Queue disabled						
1	0	0	X	NotFull ⁶	FIFO	
1	0	0	None	Full ⁷	None	Frame lost by FIFO full (FIFO Overflow)
1	0	0	Free	Full	FirstMB	
1	0	0	NotFree	Full	FirstMB	
1	0	1	None	NotFull	FIFO	
1	0	1	None	Full	None	Frame lost by FIFO full (FIFO Overflow)

Table continues on the next page...

Table 40-105. Matching possibilities and resulting reception structures (continued)

RFEN	IRMQ	MRP	Matched in MB	Matched in FIFO	Reception structure	Description
1	0	1	Free	X	FirstMB	
1	0	1	NotFree	X	FirtsMb	Overrun
FIFO enabled, Queue enabled						
1	1	0	X	NotFull	FIFO	
1	1	0	None	Full	None	Frame lost by FIFO full (FIFO Overflow)
1	1	0	Free	Full	FirstMB	
1	1	0	NotFree	Full	LastMb	Overrun
1	1	1	None	NotFull	FIFO	
1	1	1	Free	X	FirstMB	
1	1	1	NotFree	NotFull	FIFO	
1	1	1	NotFree	Full	LastMb	Overrun

1. This is a don't care condition.
2. Matched in MB "None" means that the frame has not matched any MB (free-to-receive or non-free-to-receive).
3. This is a forbidden condition.
4. Matched in MB "Free" means that the frame matched at least one MB free-to-receive regardless of whether it has matched MBs non-free-to-receive.
5. Matched in FIFO "None" means that the frame has not matched any filter in FIFO. It is as if the FIFO didn't exist (CTRL2[RFEN]=0).
6. Matched in FIFO "NotFull" means that the frame has matched a FIFO filter and has empty slots to receive it.
7. Matched in FIFO "Full" means that the frame has matched a FIFO filter but couldn't store it because it has no empty slots to receive it.

If a non-safe Mailbox inactivation (see [Mailbox inactivation](#)) occurs during matching process and the Mailbox inactivated is the temporary matching winner then the temporary matching winner is invalidated. The matching elements scan is not stopped nor restarted, it continues normally. The consequence is that the current matching process works as if the matching elements compared before the inactivation did not exist, therefore a message may be lost.

Suppose, for example, that the FIFO is disabled, IRMQ is enabled and there are two MBs with the same ID, and FlexCAN starts receiving messages with that ID. Let us say that these MBs are the second and the fifth in the array. When the first message arrives, the matching algorithm will find the first match in MB number 2. The code of this MB is EMPTY, so the message is stored there. When the second message arrives, the matching algorithm will find MB number 2 again, but it is not "free-to-receive", so it will keep looking and find MB number 5 and store the message there. If yet another message with the same ID arrives, the matching algorithm finds out that there are no matching MBs that are "free-to-receive", so it decides to overwrite the last matched MB, which is number 5. In doing so, it sets the CODE field of the MB to indicate OVERRUN.

The ability to match the same ID in more than one MB can be exploited to implement a reception queue (in addition to the full featured FIFO) to allow more time for the CPU to service the MBs. By programming more than one MB with the same ID, received messages will be queued into the MBs. The CPU can examine the Time Stamp field of the MBs to determine the order in which the messages arrived.

Matching to a range of IDs is possible by using ID Acceptance Masks. FlexCAN supports individual masking per MB. Refer to the description of the Rx Individual Mask Registers (RXIMRx). During the matching algorithm, if a mask bit is asserted, then the corresponding ID bit is compared. If the mask bit is negated, the corresponding ID bit is "don't care". Please note that the Individual Mask Registers are implemented in RAM, so they are not initialized out of reset. Also, they can only be programmed while the module is in Freeze mode; otherwise, they are blocked by hardware.

FlexCAN also supports an alternate masking scheme with only four mask registers (RXGMASK, RX14MASK, RX15MASK and RXFGMASK) for backwards compatibility with legacy applications. This alternate masking scheme is enabled when the IRMQ bit in the MCR Register is negated.

40.5.5 Move process

There are two types of move process: move-in and move-out.

40.5.5.1 Move-in

The move-in process is the copy of a message received by an Rx SMB to a Rx Mailbox or FIFO that has matched it. If the move destination is the Rx FIFO, attributes of the message are also copied to the RXFIR FIFO. Each Rx SMB has its own move-in process, but only one is performed at a given time as described ahead. The move-in starts only when the message held by the Rx SMB has a corresponding matching winner (see [Matching process](#)) and all of the following conditions are true:

- The CAN bus has reached or let past either:
 - The second bit of Intermission field next to the frame that carried the message that is in the Rx SMB
 - The first bit of an overload frame next to the frame that carried the message that is in the Rx SMB
- There is no ongoing matching process

- The destination Mailbox is not locked by the CPU
- There is no ongoing move-in process from another Rx SMB. If more than one move-in processes are to be started at the same time both are performed and the newest substitutes the oldest.

The term *pending move-in* is used throughout the documentation and stands for a move-to-be that still does not satisfy all of the aforementioned conditions.

The move-in is cancelled and the Rx SMB is able to receive another message if any of the following conditions is satisfied:

- The destination Mailbox is inactivated after the CAN bus has reached the first bit of Intermission field next to the frame that carried the message and its matching process has finished
- There is a previous pending move-in to the same destination Mailbox
- The Rx SMB is receiving a frame transmitted by the FlexCAN itself and the self-reception is disabled (MCR[SRXDIS] bit is asserted)
- Any CAN protocol error is detected

Note that the pending move-in is not cancelled if the module enters Freeze or Low-Power mode. It only stays on hold waiting for exiting Freeze and Low-Power mode and to be unlocked. If an MB is unlocked during Freeze mode, the move-in happens immediately.

The move-in process is the execution by the FlexCAN of the following steps:

1. if the message is destined to the Rx FIFO, push IDHIT into the RXFIR FIFO;
2. reads the words DATA0-3 and DATA4-7 from the Rx SMB;
3. writes it in the words DATA0-3 and DATA4-7 of the Rx Mailbox;
4. reads the words Control/Status and ID from the Rx SMB;
5. writes it in the words Control/Status and ID of the Rx Mailbox, updating the CODE field.

The move-in process is not atomic, in such a way that it is immediately cancelled by the inactivation of the destination Mailbox (see [Mailbox inactivation](#)) and in this case the Mailbox may be left partially updated, thus incoherent. The exception is if the move-in destination is an Rx FIFO Message Buffer, then the process cannot be cancelled.

The BUSY Bit (least significant bit of the CODE field) of the destination Message Buffer is asserted while the move-in is being performed to alert the CPU that the Message Buffer content is temporarily incoherent.

40.5.5.2 Move-out

The move-out process is the copy of the content from a Tx Mailbox to the Tx SMB when a message for transmission is available (see Section "Arbitration process"). The move-out occurs in the following conditions:

- The first bit of Intermission field
- During Bus Off state when TX Error Counter is in the 124 to 128 range
- During Bus Idle state
- During Wait For Bus Idle state

The move-out process is not atomic. Only the CPU has priority to access the memory concurrently out of Bus Idle state. In Bus Idle, the move-out has the lowest priority to the concurrent memory accesses.

40.5.6 Data coherence

In order to maintain data coherency and FlexCAN proper operation, the CPU must obey the rules described in [Transmit process](#) and [Receive process](#).

40.5.6.1 Transmission abort mechanism

The abort mechanism provides a safe way to request the abortion of a pending transmission. A feedback mechanism is provided to inform the CPU if the transmission was aborted or if the frame could not be aborted and was transmitted instead.

Two primary conditions must be fulfilled in order to abort a transmission:

- MCR[AEN] bit must be asserted
- The first CPU action must be the writing of abort code (0b1001) into the CODE field of the Control and Status word.

The active MBs configured as transmission must be aborted first and then they may be updated. If the abort code is written to a Mailbox that is currently being transmitted, or to a Mailbox that was already loaded into the SMB for transmission, the write operation is blocked and the MB is kept active, but the abort request is captured and kept pending until one of the following conditions are satisfied:

- The module loses the bus arbitration
- There is an error during the transmission
- The module is put into Freeze mode

- The module enters in BusOff state
- There is an overload frame

If none of the conditions above are reached, the MB is transmitted correctly, the interrupt flag is set in the IFLAG register, and an interrupt to the CPU is generated (if enabled). The abort request is automatically cleared when the interrupt flag is set. On the other hand, if one of the above conditions is reached, the frame is not transmitted; therefore, the abort code is written into the CODE field, the interrupt flag is set in the IFLAG, and an interrupt is (optionally) generated to the CPU.

If the CPU writes the abort code before the transmission begins internally, then the write operation is not blocked; therefore, the MB is updated and the interrupt flag is set. In this way the CPU just needs to read the abort code to make sure the active MB was *safely inactivated*. Although the AEN bit is asserted and the CPU wrote the abort code, in this case the MB is inactivated and not aborted, because the transmission did not start yet. One Mailbox is only aborted when the abort request is captured and kept pending until one of the previous conditions are satisfied.

The abort procedure can be summarized as follows:

- CPU checks the corresponding IFLAG and clears it, if asserted.
- CPU writes 0b1001 into the CODE field of the C/S word.
- CPU waits for the corresponding IFLAG indicating that the frame was either transmitted or aborted.
- CPU reads the CODE field to check if the frame was either transmitted (CODE=0b1000) or aborted (CODE=0b1001).
- It is necessary to clear the corresponding IFLAG in order to allow the MB to be reconfigured.

40.5.6.2 Mailbox inactivation

Inactivation is a mechanism provided to protect the Mailbox against updates by the FlexCAN internal processes, thus allowing the CPU to rely on Mailbox data coherence after having updated it, even in Normal mode.

Inactivation of transmission Mailboxes must be performed just when MCR[AEN] bit is deasserted.

If a Mailbox is inactivated, it participates in neither the arbitration process nor the matching process until it is reactivated. See [Transmit process](#) and [Receive process](#) for more detailed instruction on how to inactivate and reactivate a Mailbox.

To inactivate a Mailbox, the CPU must update its CODE field to INACTIVE (either 0b0000 or 0b1000).

Because the user is not able to synchronize the CODE field update with the FlexCAN internal processes, an inactivation can lead to undesirable results:

- A frame in the bus that matches the filtering of the inactivated Rx Mailbox may be lost without notice, even if there are other Mailboxes with the same filter
- A frame containing the message within the inactivated Tx Mailbox may be transmitted without notice

In order to eliminate such risk and perform a *safe inactivation* the CPU must use the following mechanism along with the inactivation itself:

- For Tx Mailboxes, the Transmission Abort (see [Transmission abort mechanism](#))

The inactivation automatically unlocks the Mailbox (see [Mailbox lock mechanism](#)).

NOTE

Message Buffers that are part of the Rx FIFO cannot be inactivated. There is no write protection on the FIFO region by FlexCAN. CPU must maintain data coherency in the FIFO region when RFEN is asserted.

40.5.6.3 Mailbox lock mechanism

Other than Mailbox inactivation, FlexCAN has another data coherence mechanism for the receive process. When the CPU reads the Control and Status word of an Rx MB with codes FULL or OVERRUN, FlexCAN assumes that the CPU wants to read the whole MB in an atomic operation, and therefore it sets an internal lock flag for that MB. The lock is released when the CPU reads the Free Running Timer (global unlock operation), or when it reads the Control and Status word of another MB regardless of its code. A CPU write into C/S word also unlocks the MB, but this procedure is not recommended for normal unlock use because it cancels a pending-move and potentially may lose a received message. The MB locking is done to prevent a new frame to be written into the MB while the CPU is reading it.

NOTE

The locking mechanism applies only to Rx MBs that are not part of FIFO and have a code different than INACTIVE

(0b0000) or EMPTY¹ (0b0100). Also, Tx MBs can not be locked.

Suppose, for example, that the FIFO is disabled and the second and the fifth MBs of the array are programmed with the same ID, and FlexCAN has already received and stored messages into these two MBs. Suppose now that the CPU decides to read MB number 5 and at the same time another message with the same ID is arriving. When the CPU reads the Control and Status word of MB number 5, this MB is locked. The new message arrives and the matching algorithm finds out that there are no "free-to-receive" MBs, so it decides to override MB number 5. However, this MB is locked, so the new message can not be written there. It will remain in the SMB waiting for the MB to be unlocked, and only then will be written to the MB.

If the MB is not unlocked in time and yet another new message with the same ID arrives, then the new message overwrites the one on the SMB and there will be no indication of lost messages either in the CODE field of the MB or in the Error and Status Register.

While the message is being moved-in from the SMB to the MB, the BUSY bit on the CODE field is asserted. If the CPU reads the Control and Status word and finds out that the BUSY bit is set, it should defer accessing the MB until the BUSY bit is negated.

Note

If the BUSY bit is asserted or if the MB is empty, then reading the Control and Status word does not lock the MB.

Inactivation takes precedence over locking. If the CPU inactivates a locked Rx MB, then its lock status is negated and the MB is marked as invalid for the current matching round. Any pending message on the SMB will not be transferred anymore to the MB. An MB is unlocked when the CPU reads the Free Running Timer Register (see Section "Free Running Timer Register (TIMER)"), or the C/S word of another MB.

Lock and unlock mechanisms have the same functionality in both Normal and Freeze modes.

An unlock during Normal or Freeze mode results in the move-in of the pending message. However, the move-in is postponed if an unlock occurs during a low power mode (see [Modes of operation](#)) and it will take place only when the module resumes to Normal or Freeze modes.

1. In previous FlexCAN versions, reading the C/S word locked the MB even if it was EMPTY. This behavior is maintained when the IRMQ bit is negated.

40.5.7 Rx FIFO

The Rx FIFO is receive-only and is enabled by asserting the MCR[RFEN] bit. The reset value of this bit is zero to maintain software backward compatibility with previous versions of the module that did not have the FIFO feature. The FIFO is 6-message deep. The memory region occupied by the FIFO structure (both Message Buffers and FIFO engine) is described in [Rx FIFO structure](#). The CPU can read the received messages sequentially, in the order they were received, by repeatedly reading a Message Buffer structure at the output of the FIFO.

The IFLAG[BUF5I] (Frames available in Rx FIFO) is asserted when there is at least one frame available to be read from the FIFO. An interrupt is generated if it is enabled by the corresponding mask bit. Upon receiving the interrupt, the CPU can read the message (accessing the output of the FIFO as a Message Buffer) and the RXFIR register and then clear the interrupt. If there are more messages in the FIFO the act of clearing the interrupt updates the output of the FIFO with the next message and update the RXFIR with the attributes of that message, reissuing the interrupt to the CPU. Otherwise, the flag remains negated. The output of the FIFO is only valid whilst the IFLAG[BUF5I] is asserted.

The IFLAG[BUF6I] (Rx FIFO Warning) is asserted when the number of unread messages within the Rx FIFO is increased to 5 from 4 due to the reception of a new one, meaning that the Rx FIFO is almost full. The flag remains asserted until the CPU clears it.

The IFLAG[BUF7I] (Rx FIFO Overflow) is asserted when an incoming message was lost because the Rx FIFO is full. Note that the flag will not be asserted when the Rx FIFO is full and the message was captured by a Mailbox. The flag remains asserted until the CPU clears it.

Clearing one of those three flags does not affect the state of the other two.

An interrupt is generated if an IFLAG bit is asserted and the corresponding mask bit is asserted too.

A powerful filtering scheme is provided to accept only frames intended for the target application, reducing the interrupt servicing work load. The filtering criteria is specified by programming a table of up to 128 32-bit registers, according to CTRL2[RFFN] setting, that can be configured to one of the following formats (see also [Rx FIFO structure](#)):

- Format A: 128 IDAFs (extended or standard IDs including IDE and RTR)
- Format B: 256 IDAFs (standard IDs or extended 14-bit ID slices including IDE and RTR)
- Format C: 512 IDAFs (standard or extended 8-bit ID slices)

Note

A chosen format is applied to all entries of the filter table. It is not possible to mix formats within the table.

Every frame available in the FIFO has a corresponding IDHIT (Identifier Acceptance Filter Hit Indicator) that can be read by accessing the RXFIR register. The RXFIR[IDHIT] field refers to the message at the output of the FIFO and is valid while the IFLAG[BUF5I] flag is asserted. The RXFIR register must be read only before clearing the flag, which guarantees that the information refers to the correct frame within the FIFO.

Up to 32 elements of the filter table are individually affected by the Individual Mask Registers (RXIMRx), according to the setting of CTRL2[RFFN], allowing very powerful filtering criteria to be defined. If the IRMQ bit is negated, then the FIFO filter table is affected by RXFGMASK.

40.5.8 CAN protocol related features

This section describes the CAN protocol related features.

40.5.8.1 Remote frames

Remote frame is a special kind of frame. The user can program a mailbox to be a Remote Request Frame by writing the mailbox as Transmit with the RTR bit set to '1'. After the remote request frame is transmitted successfully, the mailbox becomes a Receive Message Buffer, with the same ID as before.

When a remote request frame is received by FlexCAN, it can be treated in three ways, depending on Remote Request Storing (CTRL2[RRS]) and Rx FIFO Enable (MCR[RFEN]) bits:

- If RRS is negated the frame's ID is compared to the IDs of the Transmit Message Buffers with the CODE field 0b1010. If there is a matching ID, then this mailbox frame will be transmitted. Note that if the matching mailbox has the RTR bit set, then FlexCAN will transmit a remote frame as a response. The received remote request frame is not stored in a receive buffer. It is only used to trigger a transmission of a frame in response. The mask registers are not used in remote frame matching, and all ID bits (except RTR) of the incoming received frame should match. In the case that a remote request frame was received and matched a mailbox, this message buffer

immediately enters the internal arbitration process, but is considered as normal Tx mailbox, with no higher priority. The data length of this frame is independent of the DLC field in the remote frame that initiated its transmission.

- If RRS is asserted the frame's ID is compared to the IDs of the receive mailboxes with the CODE field 0b0100, 0b0010 or 0b0110. If there is a matching ID, then this mailbox will store the remote frame in the same fashion of a data frame. No automatic remote response frame will be generated. The mask registers are used in the matching process.
- If RFEN is asserted FlexCAN will not generate an automatic response for remote request frames that match the FIFO filtering criteria. If the remote frame matches one of the target IDs, it will be stored in the FIFO and presented to the CPU. Note that for filtering formats A and B, it is possible to select whether remote frames are accepted or not. For format C, remote frames are always accepted (if they match the ID). Remote Request Frames are considered as normal frames, and generate a FIFO overflow when a successful reception occurs and the FIFO is already full.

40.5.8.2 Overload frames

FlexCAN does transmit overload frames due to detection of following conditions on CAN bus:

- Detection of a dominant bit in the first/second bit of Intermission
- Detection of a dominant bit at the 7th bit (last) of End of Frame field (Rx frames)
- Detection of a dominant bit at the 8th bit (last) of Error Frame Delimiter or Overload Frame Delimiter

40.5.8.3 Time stamp

The value of the Free Running Timer is sampled at the beginning of the Identifier field on the CAN bus, and is stored at the end of "move-in" in the TIME STAMP field, providing network behavior with respect to time.

The Free Running Timer can be reset upon a specific frame reception, enabling network time synchronization. See the TSYN description in the description of the Control 1 Register (CTRL1).

40.5.8.4 Protocol timing

The following figure shows the structure of the clock generation circuitry that feeds the CAN Protocol Engine (PE) submodule. The clock source bit CLKSRC in the CTRL1 Register defines whether the internal clock is connected to the output of a crystal oscillator (Oscillator Clock) or to the Peripheral Clock. In order to guarantee reliable operation, the clock source should be selected while the module is in Disable Mode (bit MDIS set in the Module Configuration Register).

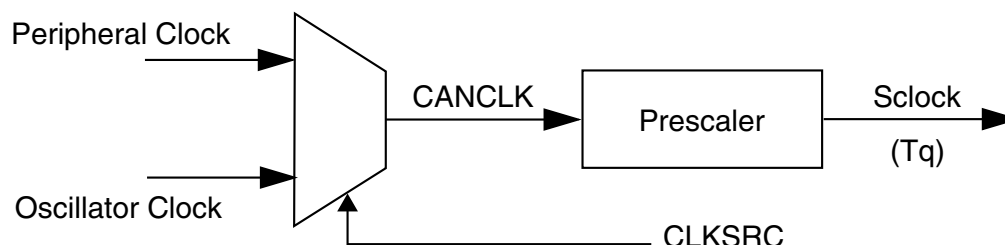


Figure 40-92. CAN engine clocking scheme

The oscillator clock should be selected whenever a tight tolerance (up to 0.1%) is required in the CAN bus timing. The oscillator clock has better jitter performance.

The FlexCAN module supports a variety of means to setup bit timing parameters that are required by the CAN protocol. The Control Register has various fields used to control bit timing parameters: PRESDIV, PROPSEG, PSEG1, PSEG2 and RJW. See the description of the Control 1 Register (CTRL1).

The PRESDIV field controls a prescaler that generates the Serial Clock (Sclock), whose period defines the 'time quantum' used to compose the CAN waveform. A time quantum is the atomic unit of time handled by the CAN engine.

$$f_{Tq} = \frac{f_{CANCLK}}{\text{Prescaler Value}}$$

A bit time is subdivided into three segments² (see [Figure 40-93](#) and [Table 40-106](#)):

- SYNC_SEG: This segment has a fixed length of one time quantum. Signal edges are expected to happen within this section

2. For further explanation of the underlying concepts, see ISO/DIS 11519–1, Section 10.3. See also the CAN 2.0A/B protocol specification for bit timing.

- Time Segment 1: This segment includes the Propagation Segment and the Phase Segment 1 of the CAN standard. It can be programmed by setting the PROPSEG and the PSEG1 fields of the CTRL1 Register so that their sum (plus 2) is in the range of 4 to 16 time quanta
- Time Segment 2: This segment represents the Phase Segment 2 of the CAN standard. It can be programmed by setting the PSEG2 field of the CTRL1 Register (plus 1) to be 2 to 8 time quanta long

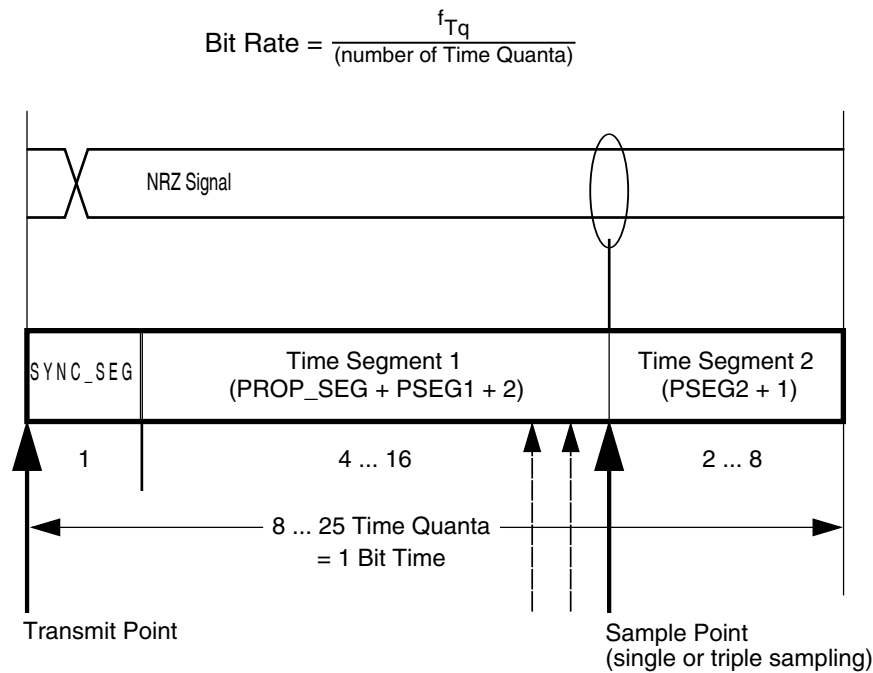


Figure 40-93. Segments within the bit time

Whenever CAN bit is used as a measure of duration (for example, MCR[FRZACK] and MCR[LPMACK]), the number of peripheral clocks in one CAN bit can be calculated as:

$$NCCP = \frac{f_{SYS} \times [1 + (PSEG1 + 1) + (PSEG2 + 1) + (PROPSEG + 1)] \times (PRES DIV + 1)}{f_{CANCLK}}$$

where:

- NCCP is the number of peripheral clocks in one CAN bit;
- f_{CANCLK} is the Protocol Engine (PE) Clock (see Figure "CAN Engine Clocking Scheme"), in Hz;
- f_{SYS} is the frequency of operation of the system (CHI) clock, in Hz;
- PSEG1 is the value in CTRL1[PSEG1] field;
- PSEG2 is the value in CTRL1[PSEG2] field;

- PROPSEG is the value in CTRL1[PROPSEG] field;
- PRESDIV is the value in CTRL1[PRESDIV] field.

For example, 180 CAN bits = 180 x NCCP peripheral clock periods.

Table 40-106. Time segment syntax

Syntax	Description
SYNC_SEG	System expects transitions to occur on the bus during this period.
Transmit Point	A node in transmit mode transfers a new value to the CAN bus at this point.
Sample Point	A node samples the bus at this point. If the three samples per bit option is selected, then this point marks the position of the third sample.

The following table gives an overview of the CAN compliant segment settings and the related parameter values.

Table 40-107. Bosch CAN 2.0B standard compliant bit time segment settings

Time segment 1	Time segment 2	Re-synchronization jump width
5 .. 10	2	1 .. 2
4 .. 11	3	1 .. 3
5 .. 12	4	1 .. 4
6 .. 13	5	1 .. 4
7 .. 14	6	1 .. 4
8 .. 15	7	1 .. 4
9 .. 16	8	1 .. 4

Note

The user must ensure the bit time settings are in compliance with the CAN Protocol standard (ISO 11898-1). For bit time calculations, use an IPT (Information Processing Time) of 2, which is the value implemented in the FlexCAN module.

40.5.8.5 Arbitration and matching timing

During normal reception and transmission of frames, the matching, arbitration, move-in and move-out processes are executed during certain time windows inside the CAN frame, as shown in the following figures.

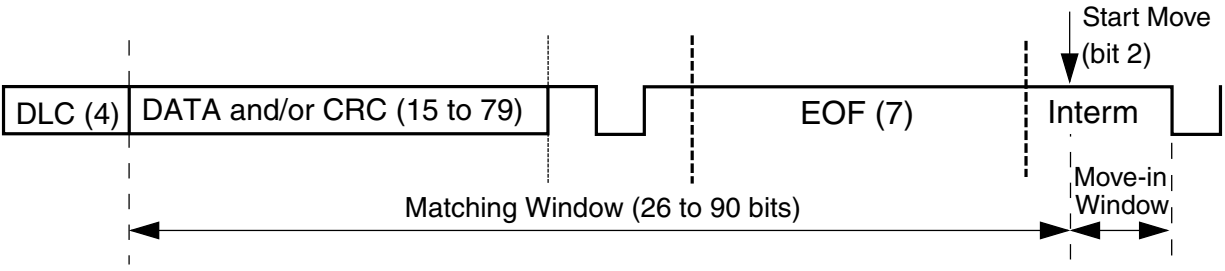


Figure 40-94. Matching and move-in time windows

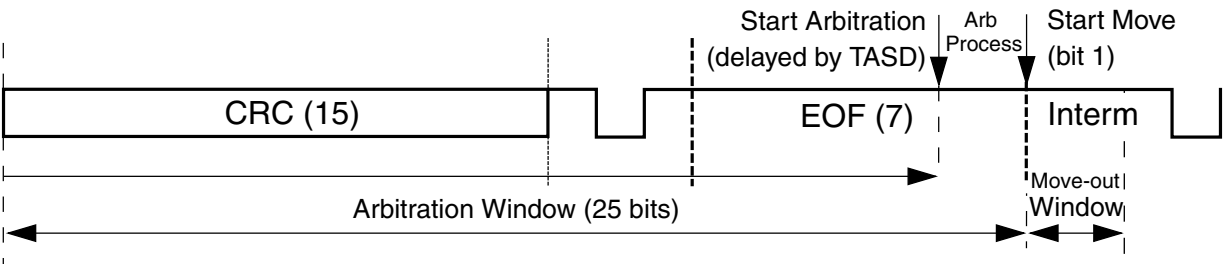


Figure 40-95. Arbitration and move-out time windows

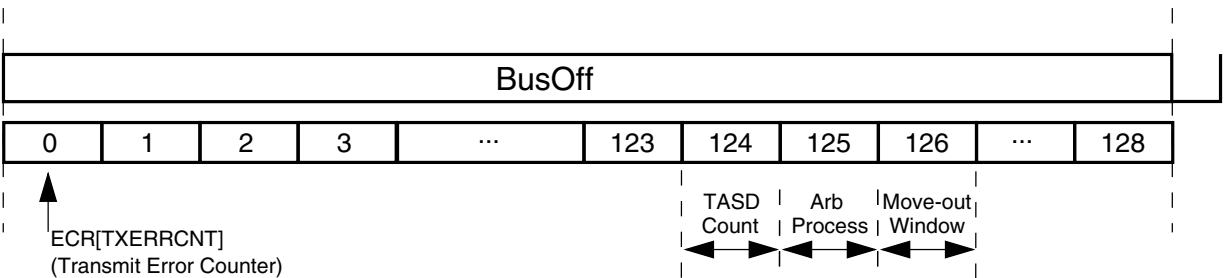


Figure 40-96. Arbitration at the end of bus off and move-out time windows

NOTE

The matching and arbitration timing shown in the preceding figures do not take into account the delay caused by the concurrent memory access due to the CPU or other internal FlexCAN sub-blocks.

When doing matching and arbitration, FlexCAN needs to scan the whole Message Buffer memory during the available time slot. In order to have sufficient time to do that, the following requirements must be observed:

- A valid CAN bit timing must be programmed, as indicated in [Table 40-107](#)
- The peripheral clock frequency can not be smaller than the oscillator clock frequency, see [Clock domains and restrictions](#)
- There must be a minimum ratio between the peripheral clock frequency and the CAN bit rate, as specified in the following table:

Table 40-108. Minimum ratio between peripheral clock frequency and CAN bit rate

Number of Message Buffers	RFEN	Minimum number of peripheral clocks per CAN bit
16 and 32	0	16
64	0	25
16	1	16
32	1	17
64	1	30

A direct consequence of the first requirement is that the minimum number of time quanta per CAN bit must be 8, therefore the oscillator clock frequency should be at least 8 times the CAN bit rate. The minimum frequency ratio specified in the preceding table can be achieved by choosing a high enough peripheral clock frequency when compared to the oscillator clock frequency, or by adjusting one or more of the bit timing parameters (PRES DIV, PROPSEG, PSEG1, PSEG2) contained in the Control 1 Register (CTRL1).

In case of synchronous operation (when the peripheral clock frequency is equal to the oscillator clock frequency), the number of peripheral clocks per CAN bit can be adjusted by selecting an adequate value for PRES DIV in order to meet the requirement in the preceding table. In case of asynchronous operation (the peripheral clock frequency greater than the oscillator clock frequency), the number of peripheral clocks per CAN bit can be adjusted by both PRES DIV and/or the frequency ratio.

As an example, taking the case of 64 MBs, if the oscillator and peripheral clock frequencies are equal and the CAN bit timing is programmed to have 8 time quanta per bit, then the prescaler factor (PRES DIV + 1) should be at least 2. For prescaler factor equal to one and CAN bit timing with 8 time quanta per bit, the ratio between peripheral and oscillator clock frequencies should be at least 2.

40.5.9 Clock domains and restrictions

The FlexCAN module has two clock domains asynchronous to each other:

- The Bus Domain feeds the Control Host Interface (CHI) submodule and is derived from the peripheral clock.
- The Oscillator Domain feeds the CAN Protocol Engine (PE) submodule and is derived directly from a crystal oscillator clock, so that very low jitter performance can be achieved on the CAN bus.

When CTRL1[CLKSRC] bit is set, synchronous operation occurs because both domains are connected to the peripheral clock (creating a 1:1 ratio between the peripheral and oscillator domain clocks).

When the two domains are connected to clocks with different frequencies and/or phases, there are restrictions on the frequency relationship between the two clock domains. In the case of asynchronous operation, the Bus Domain clock frequency must always be greater than the Oscillator Domain clock frequency.

NOTE

Asynchronous operation with a 1:1 ratio between peripheral and oscillator clocks is not allowed.

40.5.10 Modes of operation details

The FlexCAN module has functional modes and low-power modes. See [Modes of operation](#) for an introductory description of all the modes of operation. The following sub-sections contain functional details on Freeze mode and the low-power modes.

CAUTION

“Permanent Dominant” failure on CAN Bus line is not supported by FlexCAN. If a Low-Power request or Freeze mode request is done during a “Permanent Dominant”, the corresponding acknowledge can never be asserted.

40.5.10.1 Freeze mode

This mode is requested by the CPU through the assertion of the HALT bit in the MCR Register or when the chip is put into Debug mode. In both cases it is also necessary that the FRZ bit is asserted in the MCR Register and the module is not in a low-power mode. The acknowledgement is obtained through the assertion by the FlexCAN of FRZ_ACK bit in the same register. The CPU must only consider the FlexCAN in Freeze mode when both request and acknowledgement conditions are satisfied.

When Freeze mode is requested during transmission or reception, FlexCAN does the following:

- Waits to be in either Intermission, Passive Error, Bus Off or Idle state
- Waits for all internal activities like arbitration, matching, move-in and move-out to finish. A pending move-in is not taken into account.

- Ignores the Rx input pin and drives the Tx pin as recessive
- Stops the prescaler, thus halting all CAN protocol activities
- Grants write access to the Error Counters Register, which is read-only in other modes
- Sets the NOT_RDY and FRZ_ACK bits in MCR

After requesting Freeze mode, the user must wait for the FRZ_ACK bit to be asserted in MCR before executing any other action, otherwise FlexCAN may operate in an unpredictable way. In Freeze mode, all memory mapped registers are accessible, except for CTRL1[CLK_SRC] bit that can be read but cannot be written.

Exiting Freeze mode is done in one of the following ways:

- CPU negates the FRZ bit in the MCR Register
- The chip is removed from Debug Mode and/or the HALT bit is negated

The FRZ_ACK bit is negated after the protocol engine recognizes the negation of the freeze request. When out of Freeze mode, FlexCAN tries to re-synchronize to the CAN bus by waiting for 11 consecutive recessive bits.

40.5.10.2 Module Disable mode

This low power mode is normally used to temporarily disable a complete FlexCAN block, with no power consumption. It is requested by the CPU through the assertion of the MDIS bit in the MCR Register and the acknowledgement is obtained through the assertion by the FlexCAN of the LPM_ACK bit in the same register. The CPU must only consider the FlexCAN in Disable mode when both request and acknowledgement conditions are satisfied.

If the module is disabled during Freeze mode, it requests to disable the clocks to the PE and CHI sub-modules, sets the LPM_ACK bit and negates the FRZ_ACK bit.

If the module is disabled during transmission or reception, FlexCAN does the following:

- Waits to be in either Idle or Bus Off state, or else waits for the third bit of Intermission and then checks it to be recessive
- Waits for all internal activities like arbitration, matching, move-in and move-out to finish. A pending move-in is not taken into account.
- Ignores its Rx input pin and drives its Tx pin as recessive

- Shuts down the clocks to the PE and CHI sub-modules
- Sets the NOTRDY and LPMACK bits in MCR

The Bus Interface Unit continues to operate, enabling the CPU to access memory mapped registers, except the Rx Mailboxes Global Mask Registers, the Rx Buffer 14 Mask Register, the Rx Buffer 15 Mask Register, the Rx FIFO Global Mask Register. The Rx FIFO Information Register, the Message Buffers, the Rx Individual Mask Registers, and the reserved words within RAM may not be accessed when the module is in Disable Mode. Exiting from this mode is done by negating the MDIS bit by the CPU, which causes the FlexCAN to request to resume the clocks and negate the LPM_ACK bit after the CAN protocol engine recognizes the negation of disable mode requested by the CPU.

40.5.10.3 Doze mode

This is a system low power mode in which the CPU bus is kept alive and a global Doze mode request is sent to all peripherals asking them to enter low-power mode. When Doze mode is globally requested, the DOZE bit in MCR Register needs to have been asserted previously for Doze mode to be triggered. The acknowledgement is obtained through the assertion by the FlexCAN of the LPMACK bit in the same register. The CPU must only consider the FlexCAN in Doze mode when both request and acknowledgement conditions are satisfied.

If Doze mode is triggered during Freeze mode, FlexCAN requests to shut down the clocks to the PE and CHI sub-modules, sets the LPMACK bit and negates the FRZACK bit. If Doze Mode is triggered during transmission or reception, FlexCAN does the following:

- Waits to be in either Idle or Bus Off state, or else waits for the third bit of Intermission and checks it to be recessive
- Waits for all internal activities like arbitration, matching, move-in and move-out to finish. A pending move-in is not taken into account.
- Ignores its Rx input pin and drives its Tx pin as recessive
- Shuts down the clocks to the PE and CHI sub-modules
- Sets the NOTRDY and LPMACK bits in MCR

The Bus Interface Unit continues to operate, enabling the CPU to access memory mapped registers, except the Rx Mailboxes Global Mask Registers, the Rx Buffer 14 Mask Register, the Rx Buffer 15 Mask Register, the Rx FIFO Global Mask Register. The Rx FIFO Information Register, the Message Buffers, the Rx Individual Mask Registers, and the reserved words within RAM may not be accessed when the module is in Doze Mode.

Exiting Doze mode is done in one of the following ways:

- CPU removing the Doze mode request
- CPU negating the DOZE bit of the MCR Register

40.5.10.4 Stop mode

This is a system low-power mode in which all chip clocks can be stopped for maximum power savings. The Stop mode is globally requested by the CPU and the acknowledgement is obtained through the assertion by the FlexCAN of a Stop Acknowledgement signal. The CPU must only consider the FlexCAN in Stop mode when both request and acknowledgement conditions are satisfied.

If FlexCAN receives the global Stop mode request during Freeze mode, it sets the LPMACK bit, negates the FRZACK bit and then sends the Stop Acknowledge signal to the CPU, in order to shut down the clocks globally.

If Stop mode is requested during transmission or reception, FlexCAN does the following:

- Waits to be in either Idle or Bus Off state, or else waits for the third bit of Intermission and checks it to be recessive
- Waits for all internal activities like arbitration, matching, move-in and move-out to finish. A pending move-in is not taken into account.
- Ignores its Rx input pin and drives its Tx pin as recessive
- Sets the NOTRDY and LPMACK bits in MCR
- Sends a Stop Acknowledge signal to the CPU, so that it can shut down the clocks globally

Stop mode is exited when the CPU resumes the clocks and removes the Stop Mode request.

After the CAN protocol engine recognizes the negation of the Stop mode request, the FlexCAN negates the LPMACK bit. FlexCAN will then wait for 11 consecutive recessive bits to synchronize to the CAN bus. As a consequence, it will not receive the frame that woke it up.

40.5.11 Interrupts

The module has many interrupt sources: interrupts due to message buffers and interrupts due to the ORed interrupts from MBs, Bus Off, Error, ECC Host Access Non-correctable Error, ECC FlexCAN Access Non-correctable Error, ECC Correctable Error, Tx Warning, and Rx Warning.

Each one of the message buffers can be an interrupt source, if its corresponding IMASK bit is set. There is no distinction between Tx and Rx interrupts for a particular buffer, under the assumption that the buffer is initialized for either transmission or reception. Each of the buffers has assigned a flag bit in the IFLAG registers. The bit is set when the corresponding buffer completes a successful transfer and is cleared when the CPU writes it to 1 (unless another interrupt is generated at the same time).

Note

It must be guaranteed that the CPU clears only the bit causing the current interrupt. For this reason, bit manipulation instructions (BSET) must not be used to clear interrupt flags. These instructions may cause accidental clearing of interrupt flags which are set after entering the current interrupt service routine.

If the Rx FIFO is enabled ($\text{MCR}[\text{RFEN}] = 1$), the interrupts corresponding to MBs 0 to 7 have different meanings. Bit 7 of IFLAG1 becomes the "FIFO Overflow" flag; bit 6 becomes the FIFO Warning flag, bit 5 becomes the "Frames Available in FIFO flag," and bits 4-0 are unused. See the description of IFLAG1 for more information.

For a combined interrupt where multiple MB interrupt sources are ORed together, the interrupt is generated when any of the associated MBs (or FIFO, if applicable) generates an interrupt. In this case, the CPU must read the IFLAG registers to determine which MB or FIFO source caused the interrupt.

The interrupt sources for Bus Off, Error, Tx Warning, and Rx Warning generate interrupts like the MB interrupt sources, and they can be read from ESR1 and ESR2. The Bus Off, Error, Tx Warning, and Rx Warning interrupt mask bits are located in CTRL1.

The interrupt sources for ECC errors—Host Access with Non-correctable Interrupt Flag, FlexCAN Access with Non-correctable Interrupt Flag, and Correctable Error Interrupt Flag—can be read in ERRSR and the corresponding interrupts masks bits are located in MECR.

40.5.12 Bus interface

The CPU access to FlexCAN registers are subject to the following rules:

- Unrestricted read and write access to supervisor registers (registers identified with S/U in Table "Module Memory Map" in Supervisor Mode or with S only) results in access error.
- Read and write access to implemented reserved address space results in access error.
- Write access to positions whose bits are all currently read-only results in access error. If at least one of the bits is not read-only then no access error is issued. Write permission to positions or some of their bits can change depending on the mode of operation or transitory state. Refer to register and bit descriptions for details.
- Read and write access to unimplemented address space results in access error.
- Read and write access to RAM located positions during Low Power Mode results in access error.
- If MAXMB is programmed with a value smaller than the available number of MBs, then the unused memory space can be used as general purpose RAM space. Note that reserved words within RAM cannot be used. As an example, suppose FlexCAN is configured with 16 MBs, RFFN is 0x0, and MAXMB is programmed with zero. The maximum number of MBs in this case becomes one. The RAM starts at 0x0080, and the space from 0x0080 to 0x008F is used by the one MB. The memory space from 0x0090 to 0x017F is available. The space between 0x0180 and 0x087F is reserved. The space from 0x0880 to 0x0883 is used by the one Individual Mask and the available memory in the Mask Registers space would be from 0x0884 to 0x08BF. From 0x08C0 through 0x09DF there are reserved words for internal use which cannot be used as general purpose RAM. As a general rule, free memory space for general purpose depends only on MAXMB.

40.5.13 Detection and Correction of Memory Errors

FlexCAN supports detection and correction of errors in memory read accesses. Each byte of FlexCAN memory is associated to 5 parity bits and the error correction mechanism assures that in this 13-bit word, errors in one bit can be corrected (corrected errors) and errors in 2 bits can be detected but not corrected (non-corrected errors). Errors in more than 2 bits may not be detected. When read, the parity bits are used to calculate a syndrome, which indicates the error in each byte.

Memory errors are indicated to Host through status register (Error Status Register (ERRSR)) and bus transfer errors, and reported through report registers (Error Report Address Register (RERRAR), Error Report Data Register (RERRDR) and Error Report Syndrome Register (RERRSYNR)).

The error detection and correction mechanism can be activated or not, controlled by ECCDIS bit in Memory Error Control Register (MECR). When disabled, updates on indications and reporting registers are stopped, but the parity bits are still calculated and written along with data in memory write operations. It assures that memory has consistent parity bits associated to data.

NOTE

All FlexCAN memory must be initialized before starting its operation in order to have the parity bits in memory properly updated. The WRMFRZ bit in Control 2 Register (CTRL2) grants write access to all memory positions from 0x080 to 0xADF.

To avoid that critical error correction configuration be accidentally changed, this protocol must be followed to enable the update of the Memory Error Control Register (MECR):

1. By default, ECRWRE bit in Control 2 Register (CTRL2) is 0 and ECRWRDIS bit in Memory Error Control Register (MECR) is 1.
2. Set ECRWRE bit in Control 2 Register (CTRL2).
3. Clear ECRWRDIS bit.
4. All writes to Memory Error Control Register (MECR) must keep ECRWRDIS cleared.
5. After configuration is done, lock the Memory Error Control Register (MECR) by either setting ECRWRDIS or clearing ECRWRE.

40.5.13.1 Sources of the Memory Access

The FlexCAN memory can be accessed by 2 major sources (or requestors):

- by Host (CPU): largest word accessed is 32-bit wide
- by FlexCAN internal processes (Match, Arbitration, RxMove on reception, TxMove on transmission): largest word accessed is 64-bit wide

The way that non-correctable errors are indicated and reported depends on the source of access.

40.5.13.2 Error Indication

Memory errors are indicated by flags HANCEIF, FANCEIF, CEIF in the Error Status Register (ERRSR). Non-correctable errors detected in memory reads requested by Host are indicated separately than the ones detected in requests by FlexCAN internal processes. Corrected errors indication makes no distinction of the source of the access. There are 3 independent flags for these 3 cases, and each flag will raise an interrupt unless it is masked by mask bits in Memory Error Control Register (MECR). If both non-corrected and corrected errors are found in different bytes in the same read operation both flags are set.

The non-corrected error detected in Host accesses are also indicated as a bus transfer error. A bus wait request may be asserted to extend the memory transaction to the moment the report registers are updated. This indication cannot be masked. If the flag bit HANCEIF is not masked, the same non-corrected error will raise a bus transfer error and an interrupt request.

Each indication flag has one Overrun flag in Error Status Register (ERRSR). The Overrun flags do not request interrupts. Overrun flags for non-correctable errors indicate that other errors of the same nature were detected after current error being treated, while overrun flags for correctable errors indicate that other errors of the same nature were detected before the current error being treated. This is the recommended handling sequence for error indication:

1. Get error report information from report registers
2. Use this information to take proper measures in the application
3. Clear the Interrupt flag
4. If the Overrun flag is active:
 - a. Alert application that at least one error could not be handled
 - b. Clear the Overrun flag

The FlexCAN internal processes can access memory in transactions larger than 32-bits. For the indication, this kind of access is considered a consecutive sequence of 32-bit accesses. If errors are found in 2 or more 32-bit words the Interrupt and Overrun flags are set simultaneously.

40.5.13.3 Error Reporting

Report registers Error Report Address Register (RERRAR), Error Report Data Register (RERRDR) and Error Report Syndrome Register (RERRSYNR) provide detailed information about the address read, raw data and syndrome read with error and indicated by the flags described in the “Error Indication” section. The address, data and syndrome registers are updated simultaneously along with the error flags, according to these rules.

1. If any of the 2 non-corrected error flags is currently set, does not update the registers (preserve the old non-corrected error reporting)
2. Else (no error flag is currently set or only the corrected error flag is currently set), update the report registers according to the new error; or according to the most severe of new errors if non-corrected and corrected errors are simultaneously detected

Reporting of errors detected in accesses larger than 32-bit follows the semantic described in the “Error Indication” section and in the Error Report Address Register (RERRAR) description.

The address reported in RERRAR and defined in ERRIAR are not the same listed in the module memory map. How the reported addresses correspond to locations in the module memory map is shown in the Error Injection Address Register (ERRIAR) description.

Addresses reported when reading memory portions organized as FIFOs, such as the Rx FIFO Structure and the Rx FIFO Information Register (RXFIR), refer to the address of the element accessed in the FIFO, not to the FIFO base address.

To assure coherence of the error report registers it is necessary to turn off the report update by clearing the MECCR[RERRDIS] bit before reading the report registers.

40.5.13.4 Response to Errors

Correctable errors have no consequence on FlexCAN operation, once the corrected data is used by Host or FlexCAN internal processes. For host-initiated reads, a non-corrected error detected does not affect FlexCAN operation.

Non-corrected errors detected on memory reads requested by FlexCAN do not stop the FlexCAN processing and this may result in incorrect operation.

If a non-corrected error occurs during a reception process (Match or RxMove), an incorrect destination may be selected to store the incoming frame, a corrupted frame may be stored in the correct destination, or both. If a non-corrected error occurs during either

an Arbitration or a TxMove process, either a non-highest priority frame may be mistakenly selected to be transmitted or no frame may be transmitted at all, when $\text{MECR}[\text{NCEFAFRZ}] = 1$.

The error report registers can provide information to the application for a customized handling of these situations. If the delay of this handling is not accepted and $\text{MECR}[\text{NCEFAFRZ}] = 1$, the FlexCAN stops operation automatically and enters Freeze mode when these errors are detected. See the Memory Error Control Register (MECR) description for more details.

40.5.13.5 Error Injection

The error injection registers ERRIAR , ERRIDPR , and ERRIPPR are used to inject errors in memory reads in order to force errors and consequently update the indication and reporting registers. How the error injection addresses correspond to locations in the module memory map is shown in the ERRIAR description.

The injection is done by flipping the data and parity bits correspondent to the bits in 1 in ERRIDPR and ERRIPPR . Injection can be selected specifically for memory accesses requested by Host or by FlexCAN internal processes.

In case of accesses larger than 32-bits, the EXTERRIE bit in Memory Error Control Register (MECR) extends the injection pattern, replicating it in 32-bit words to fill the width of the access.

NOTE

It is very unlikely, but error injection may correct a bit with error. This will not raise the error flags and reports as expected.

To ensure coherence among error injection registers and avoid spurious error injections, the HAERRIE and FAERRIE bits in Memory Error Control Register (MECR) must be cleared while configuring the memory injection registers.

40.6 Initialization/application information

This section provide instructions for initializing the FlexCAN module.

40.6.1 FlexCAN initialization sequence

The FlexCAN module may be reset as follows:

- Chip level hard reset, which resets all memory mapped registers asynchronously
- SOFTRST bit in MCR, which resets some of the memory mapped registers synchronously. See [Table 40-2](#) to see what registers are affected by soft reset.

Soft reset is synchronous and has to follow an internal request/acknowledge procedure across clock domains. Therefore, it may take some time to fully propagate its effects. The SOFTRST bit remains asserted while soft reset is pending, so software can poll this bit to know when the reset has completed. Also, soft reset can not be applied while clocks are shut down in a low power mode. The low power mode should be exited and the clocks resumed before applying soft reset.

The clock source (CLKSRC bit) should be selected while the module is in Disable mode. After the clock source is selected and the module is enabled (MDIS bit negated), FlexCAN automatically goes to Freeze mode. In Freeze mode, FlexCAN is unsynchronized to the CAN bus, the HALT and FRZ bits in MCR Register are set, the internal state machines are disabled and the FRZACK and NOTRDY bits in the MCR Register are set. The Tx pin is in recessive state and FlexCAN does not initiate any transmission or reception of CAN frames. Note that the Message Buffers and the Rx Individual Mask Registers are not affected by reset, so they are not automatically initialized.

For any configuration change/initialization it is required that FlexCAN is put into Freeze mode; see [Freeze mode](#). The following is a generic initialization sequence applicable to the FlexCAN module:

- Initialize the Module Configuration Register
 - Enable the individual filtering per MB and reception queue features by setting the IRMQ bit
 - Enable the warning interrupts by setting the WRNEN bit
 - If required, disable frame self reception by setting the SRXDIS bit
 - Enable the Rx FIFO by setting the RFEN bit
 - Enable the abort mechanism by setting the AEN bit
 - Enable the local priority feature by setting the LPRIOEN bit
- Initialize the Control Register
 - Determine the bit timing parameters: PROPSEG, PSEG1, PSEG2, RJW

- Determine the bit rate by programming the PRES DIV field
- Determine the internal arbitration mode (LBUF bit)
- Initialize the Message Buffers
 - The Control and Status word of all Message Buffers must be initialized
 - If Rx FIFO was enabled, the ID filter table must be initialized
 - Other entries in each Message Buffer should be initialized as required
- Initialize the Rx Individual Mask Registers
- When ECC is enabled, follow the initialization note in [Detection and Correction of Memory Errors](#)
- Set required interrupt mask bits in the IMASK Registers (for all MB interrupts) and in CTRL Register (for Bus Off and Error interrupts)
- Negate the HALT bit in MCR

Starting with the last event, FlexCAN attempts to synchronize to the CAN bus.

Chapter 41

Modular CAN (M_CAN)

41.1 Chip-specific M_CAN information

41.1.1 M_CAN Message RAM allocation

On this chip, each M_CAN instance can address 1216 words in the Message RAM.

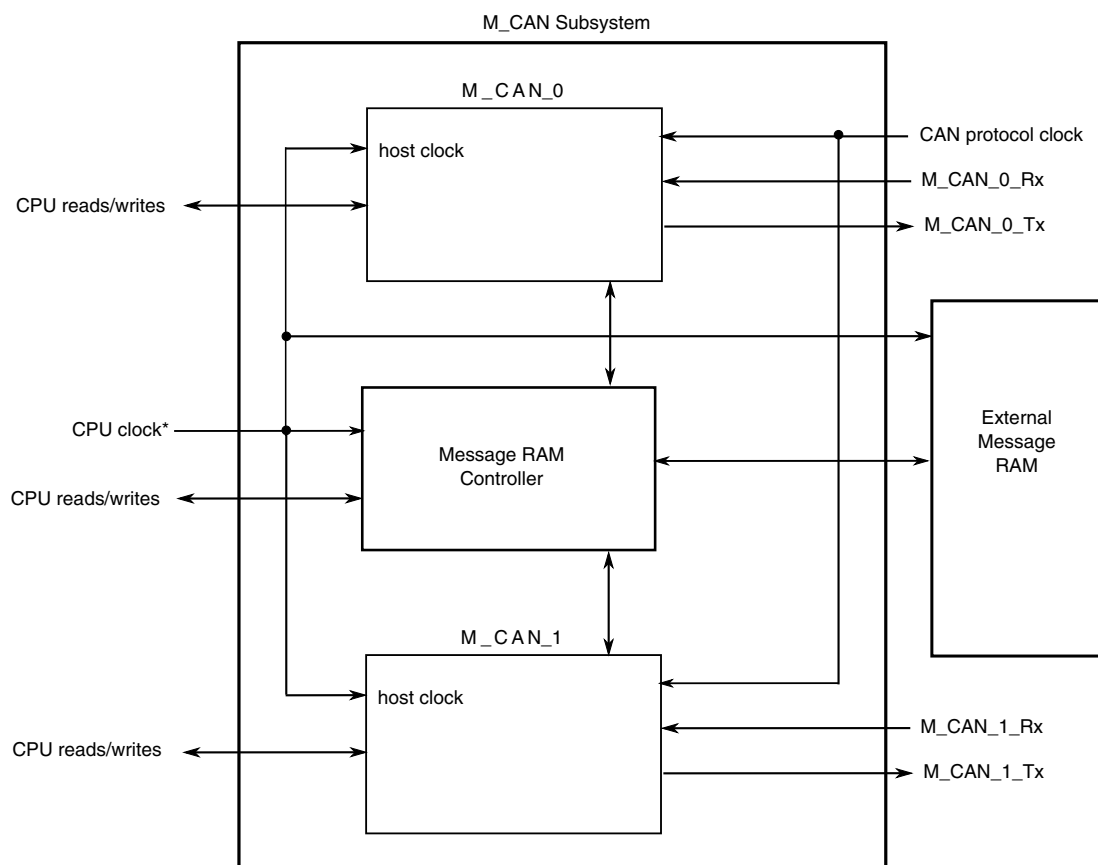
As a result, the Message RAM shared by the two M_CAN instances supports 2432 words, or 9.5 KB.

41.1.2 Introduction

The M_CAN subsystem includes:

- Two M_CAN modules
- A Message RAM controller

The M_CAN subsystem block diagram is shown in the following figure.



*Refer to the Clocking chapter for M_CAN clock details.

Figure 41-1. M_CAN subsystem block diagram

41.1.3 Functional Description

41.1.3.1 Message RAM Controller

The Message RAM Controller has the arbiter for the accesses to the external Message RAM and the ECC (Error Code Correction) Controller for the external Message RAM data.

41.1.3.1.1 Message RAM Arbiter

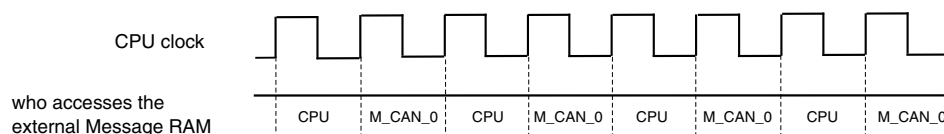
The Message RAM Arbiter is a dynamic round robin arbiter that selects which request is sent to the external Message RAM. These requests are made by the CPU, M_CAN_0, or M_CAN_1.

This arbiter ensures:

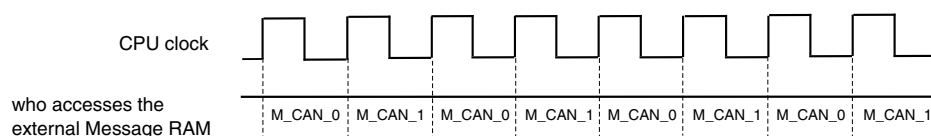
- 50% bandwidth for CPU accesses to the external Message RAM. The CPU does not wait for more than one clock cycle to access the external Message RAM (see examples 1 and 3).
- 50% bandwidth is shared between M_CAN_0 and M_CAN_1 accesses to the external Message RAM. Each M_CAN waits at least one clock cycle to access the external Message RAM (see examples 1, 2, and 3).
- If there is no CPU request, all bandwidth is distributed to M_CAN_0 and M_CAN_1 (see example 2).
- If there are no M_CAN_0 and M_CAN_1 requests, all bandwidth is distributed to the CPU.
- If there are requests from only one M_CAN, the other M_CAN's bandwidth is distributed to the first M_CAN (see example 1).

The following examples illustrate the dynamic arbiter scheme.

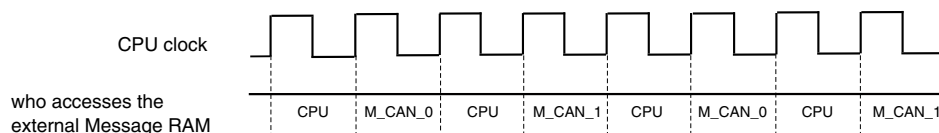
- **Example 1:** The following figure shows who accesses the external Message RAM when only CPU and M_CAN_0 try to access to it.



- **Example 2:** The following figure shows who accesses the external Message RAM when only M_CAN_0 and M_CAN_1 try to access to it.



- **Example 3:** The following figure shows who accesses the external Message RAM when the CPU, M_CAN_0, and M_CAN_1 try to access to it.



The read or write accesses to the external Message RAM use two clock cycles. In the first clock cycle, the address is available, and in the second, the data is available.

The arbiter has a pseudo address pre-fetching mechanism that allows the data of the previous access to overlap with the address of the current access. The pseudo address pre-fetching scheme saves multiple clock cycles when there are multiples accesses to the external Message RAM.

41.1.3.1.2 ECC Controller

The ECC Controller provides Single Error Correction / Double Error Detection (SECEDED). It guarantees single bit error correction and double bit error detection (without correction). The SECEDED code is not guaranteed to detect more than two bits with error.

Each 32 data bits of the external Message RAM is associated with 7 ECC bits. If all these 39 bits are zero or one, then it is flagged as non-correctable error.

For writes to the external Message RAM, the ECC bits (7-bit) are calculated using the data bits (32-bit). The data bits plus ECC bits (39-bit) are written into the specified memory address. The error detection and correction are performed on the reads from the external Message RAM.

When an M_CAN accesses the external Message RAM, the ECC bits are calculated by the ECC Controller and they are sent to this M_CAN.

41.1.3.2 External Message RAM

The external Message RAM supports only 32-bit write and read accesses.

The CPU can access the external Message RAM through the M_CAN subsystem. In this case, the CPU can do 8/16/32-bit read accesses to the external Message RAM.

41.1.3.3 Transfer Error

The M_CAN subsystem does not report any transfer error.

41.1.4 External Signals

The M_CAN subsystem external signals are shown in the following table.

Table 41-1. M_CAN subsystem external signals

Signal name	Direction	Description
M_CAN_0_Rx	input	M_CAN_0 CAN Rx signal
M_CAN_0_Tx	output	M_CAN_0 CAN Tx signal
M_CAN_1_Rx	input	M_CAN_1 CAN Rx signal
M_CAN_1_Tx	output	M_CAN_1 CAN Tx signal

41.2 Overview

The M_CAN module is the new CAN Communication Controller IP-module that can be integrated as stand-alone device or as part of an ASIC. It is described in VHDL on RTL level, prepared for synthesis. The M_CAN performs communication according to ISO11898-1 (Bosch CAN specification 2.0 part A,B) and to Bosch CAN FD specification V1.0. Additional transceiver hardware is required for connection to the physical layer.

The message storage is intended to be a single- or dual-ported Message RAM outside of the module. It is connected to the M_CAN via the Generic Master Interface. Depending on the chosen ASIC integration, multiple M_CAN controllers can share the same Message RAM.

All functions concerning the handling of messages are implemented by the Rx Handler and the Tx Handler. The Rx Handler manages message acceptance filtering, the transfer of received messages from the CAN Core to the Message RAM as well as providing receive message status information. The Tx Handler is responsible for the transfer of transmit messages from the Message RAM to the CAN Core as well as providing transmit status information.

Acceptance filtering is implemented by a combination of up to 128 filter elements where each one can be configured as a range, as a bit mask, or as a dedicated ID filter.

The M_CAN can be connected to a wide range of Host CPUs via its 8/16/32-bit Generic Slave Interface. The M_CAN's clock domain concept allows the separation between the high precision CAN clock and the Host clock, which may be generated by an FM-PLL.

41.2.1 Features

The following are the features of M_CAN.

- Conforms with CAN protocol version 2.0 part A, B and ISO 11898-1

- CAN FD with up to 64 data bytes supported
- CAN Error Logging
- AUTOSAR support
- SAE J1939 support
- Improved acceptance filtering
- Two configurable Receive FIFOs
- Separate signalling on reception of High Priority Messages
- Up to 64 dedicated Receive Buffers
- Up to 32 dedicated Transmit Buffers
- Configurable Transmit FIFO
- Configurable Transmit Queue
- Configurable Transmit Event FIFO
- Direct Message RAM access for Host CPU
- Multiple M_CANs may share the same Message RAM
- Programmable loop-back test mode
- Maskable module interrupts
- 8/16/32 bit Generic Slave Interface for connection customer-specific Host CPUs
- Two clock domains (CAN clock and Host clock)
- Power-down support
- Debug on CAN support

41.2.2 Block Diagram

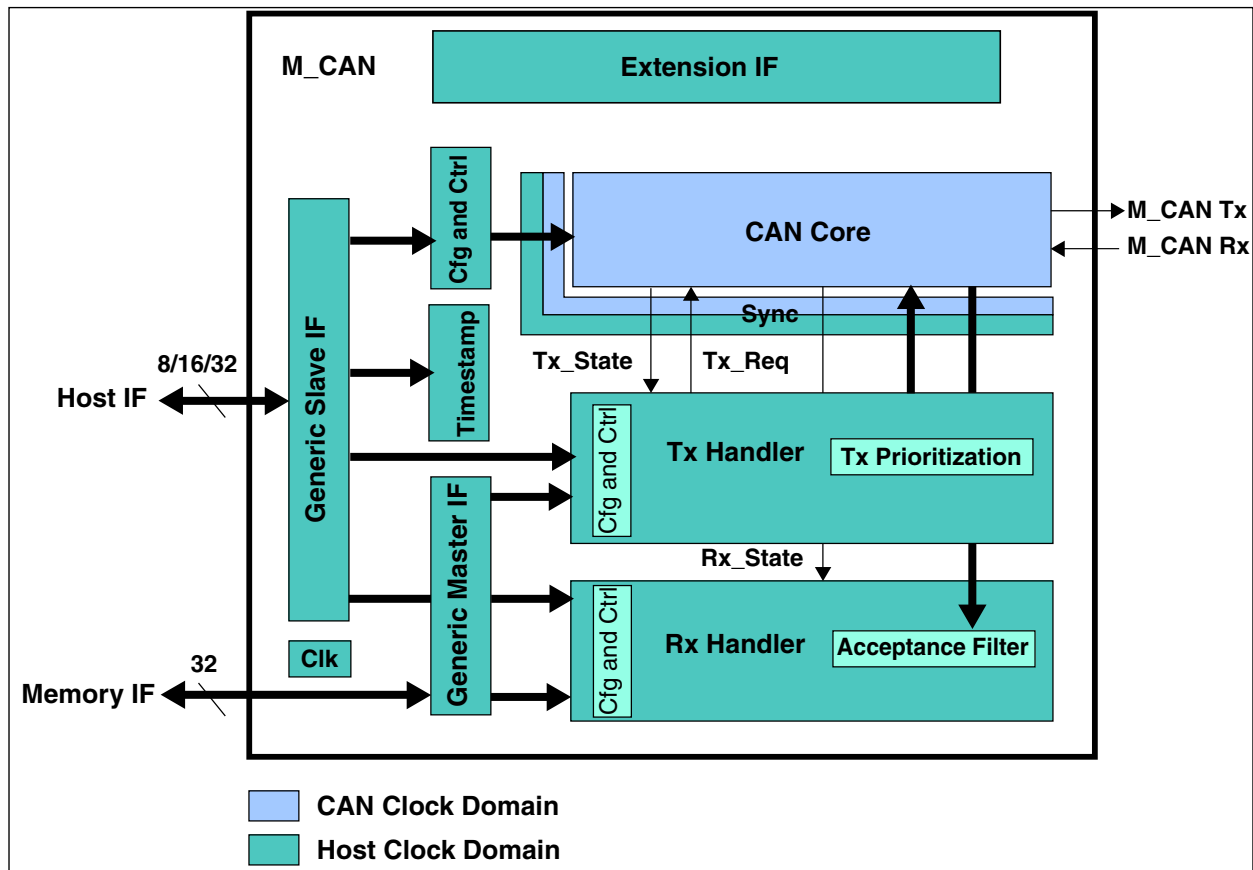


Figure 41-2. M_CAN Block Diagram

- **CAN Core:** CAN Protocol Controller and Rx/Tx Shift Register. Handles all ISO 11898-1 protocol functions. Supports 11-bit and 29-bit identifiers.
- **Sync:** Synchronizes signals from the Host clock domain to the CAN clock domain and vice versa.
- **Clk:** Synchronizes reset signal to the Host clock domain and to the CAN clock domain.
- **Cfg and Ctrl:** CAN Core related configuration and control bits.
- **Interrupt and Timestamp:** Interrupt control and 16-bit CAN bit time counter for receive and transmit timestamp generation.
- **Tx Handler:** Controls the message transfer from the external Message RAM to the CAN Core. A maximum of 32 Tx Buffers can be configured for transmission. Tx buffers can be used as dedicated Tx Buffers, as Tx FIFO, part of a Tx Queue, or as a combination of them. A Tx Event FIFO stores Tx timestamps together with the corresponding Message ID. Transmit cancellation is also supported.

- **Rx Handler:** Controls the transfer of received messages from the CAN Core to the external Message RAM. The Rx Handler supports two Receive FIFOs, each of configurable size, and up to 64 dedicated Rx Buffers for storage of all messages that have passed acceptance filtering. A dedicated Rx Buffer, in contrast to a Receive FIFO, is used to store only messages with a specific identifier. An Rx timestamp is stored together with each message. Up to 128 filters can be defined for 11-bit IDs and up to 64 filters for 29-bit IDs.
- **Connects the M_CAN to a customer specific Host CPU.** The Generic Slave Interface is capable to connect to an 8/16/32-bit bus to support a wide range of interconnection structures.
- **Generic Master Interface:** Connects the M_CAN access to an external 32-bit Message RAM. The maximum Message RAM size is 16 KB × 32-bit. A single M_CAN can use at most 4.25K × 32 bit.
- **All flags from the Interrupt Register IR as well as selected internal status and control signals are routed to this interface.** The interface is intended for connection of the M_CAN to a module-external interrupt unit or to other module-external components. The connection of these signals is optional.

41.2.3 Dual Clock Sources

To improve the EMC behavior, a spread spectrum clock can be used for the Host clock domain. Due to the high precision clocking requirements of the CAN Core, a separate clock without any modulation has to be provided as CAN clock.

Within the M_CAN module there is a synchronization mechanism implemented to ensure save data transfer between the two clock domains.

Note

In order to achieve a stable function of the M_CAN, the Host clock must always be faster than or equal to the CAN clock. Also, the modulation depth of a spread spectrum clock has to be regarded.

41.2.4 Dual Interrupt Lines

The module provides two interrupt lines. Interrupts can be routed either to M_CAN interrupt 0 or to M_CAN interrupt 1. By default all interrupts are routed to interrupt line M_CAN interrupt 0. By programming ILE[EINT0] and ILE[EINT1], the interrupt lines can be enabled or disabled separately.

41.3 Memory Map and Register Description

After hardware reset, the registers of the M_CAN hold the reset values listed in Table 1. Additionally the Bus_Off state is reset and the output M_CAN Tx is set to recessive (HIGH). The value 0x0001 (CCCR[INIT] = 1) in the CC Control Register enables software initialization. The M_CAN does not influence the CAN bus until the CPU resets CCCR[INIT] to 0.

The M_CAN module allocates an address space of 256 bytes. All registers are organized as 32-bit registers. The M_CAN is accessible by the Host CPU via the Generic Slave Interface using a data width of 8 bit (byte access), 16 bit (half-word access), or 32 bit (word access). Write access by the Host CPU to registers/bits marked with “P=Protected Write” is possible only with CCCR[CCE]=1 AND CCCR[INIT]=1. There is a delay from writing to a command register until the update of the related status register bits due to clock domain crossing.

CAUTION

In case the application software wants to access one of the reserved addresses in the M_CAN register map (read or write access), interrupt flag IR[ARA] is set, and if enable the interrupt is signalled via the assigned interrupt line (m_can_int0 or m_can_int1).

M_CAN memory map

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
0	Core Release Register (M_CAN_CREL)	32	R	See section	41.3.1/2049
4	Endian Register (M_CAN_ENDN)	32	R	8765_4321h	41.3.2/2050
C	Data Bit Timing and Prescaler Register (M_CAN_DBTP)	32	R/W	0000_0A33h	41.3.3/2051
10	Test Register (M_CAN_TEST)	32	R/W	See section	41.3.4/2052
14	RAM Watchdog Register (M_CAN_RWD)	32	R/W	0000_0000h	41.3.5/2054
18	CC Control Register (M_CAN_CCCR)	32	R/W	0000_0001h	41.3.6/2055
1C	Nominal Bit Timing and Prescaler Register (M_CAN_NBTP)	32	R/W	0600_0A03h	41.3.7/2057

Table continues on the next page...

M_CAN memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
20	Timestamp Counter Configuration Register (M_CAN_TSCC)	32	R/W	0000_0000h	41.3.8/2059
24	Timestamp Counter Value Register (M_CAN_TSCV)	32	w1c	0000_0000h	41.3.9/2060
28	Timeout Counter Configuration Register (M_CAN_TOCC)	32	R/W	FFFF_0000h	41.3.10/2060
2C	Timeout Counter Value Register (M_CAN_TOCV)	32	w1c	0000_FFFFh	41.3.11/2061
40	Error Counter Register (M_CAN_ECR)	32	R	0000_0000h	41.3.12/2062
44	Protocol Status Register (M_CAN_PSR)	32	R	0000_0707h	41.3.13/2063
48	Transmitter Delay Compensation Register (M_CAN_TDCR)	32	R/W	0000_0000h	41.3.14/2066
50	Interrupt Register (M_CAN_IR)	32	R/W	0000_0000h	41.3.15/2067
54	Interrupt Enable Register (M_CAN_IE)	32	R/W	0000_0000h	41.3.16/2070
58	Interrupt Line Select Register (M_CAN_ILS)	32	R/W	0000_0000h	41.3.17/2073
5C	Interrupt Line Enable Register (M_CAN_ILE)	32	R/W	0000_0000h	41.3.18/2077
80	Global Filter Configuration Register (M_CAN_GFC)	32	R/W	0000_0000h	41.3.19/2078
84	Standard ID Filter Configuration Register (M_CAN_SIDFC)	32	R/W	0000_0000h	41.3.20/2079
88	Extended ID Filter Configuration Register (M_CAN_XIDFC)	32	R/W	0000_0000h	41.3.21/2080
90	Extended ID and Mask Register (M_CAN_XIDAM)	32	R/W	1FFF_FFFFh	41.3.22/2080
94	High Priority Message Status Register (M_CAN_HPMS)	32	R	0000_0000h	41.3.23/2081
98	New Data 1 Register (M_CAN_NDAT1)	32	R/W	0000_0000h	41.3.24/2082
9C	New Data 2 Register (M_CAN_NDAT2)	32	R/W	0000_0000h	41.3.25/2082
A0	Rx FIFO 0 Configuration Register (M_CAN_RXF0C)	32	R/W	0000_0000h	41.3.26/2083
A4	Rx FIFO 0 Status Register (M_CAN_RXF0S)	32	R	0000_0000h	41.3.27/2084
A8	Rx FIFO 0 Acknowledge Register (M_CAN_RXF0A)	32	R/W	0000_0000h	41.3.28/2085
AC	Rx Buffer Configuration Register (M_CAN_RXBC)	32	R/W	0000_0000h	41.3.29/2085
B0	Rx FIFO 1 Configuration Register (M_CAN_RXF1C)	32	R/W	0000_0000h	41.3.30/2086

Table continues on the next page...

M_CAN memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
B4	Rx FIFO 1 Status Register (M_CAN_RXF1S)	32	R	0000_0000h	41.3.31/2087
B8	Rx FIFO 1 Acknowledge Register (M_CAN_RXF1A)	32	R/W	0000_0000h	41.3.32/2088
BC	Rx Buffer / FIFO Element Size Configuration Register (M_CAN_RXESC)	32	R/W	0000_0000h	41.3.33/2089
C0	Tx Buffer Configuration Register (M_CAN_TXBC)	32	R/W	0000_0000h	41.3.34/2091
C4	Tx FIFO/Queue Status Register (M_CAN_TXFQS)	32	R	0000_0000h	41.3.35/2092
C8	Tx Buffer Element Size Configuration (M_CAN_TXESC)	32	R/W	0000_0000h	41.3.36/2093
CC	Tx Buffer Request Pending Register (M_CAN_TXBRP)	32	R	0000_0000h	41.3.37/2094
D0	Tx Buffer Add Request Register (M_CAN_TXBAR)	32	R/W	0000_0000h	41.3.38/2095
D4	Tx Buffer Cancellation Request Register (M_CAN_TXBCR)	32	R/W	0000_0000h	41.3.39/2095
D8	Tx Buffer Transmission Occurred Register (M_CAN_TXBTO)	32	R	0000_0000h	41.3.40/2096
DC	Tx Buffer Cancellation Finished Register (M_CAN_TXBCF)	32	R	0000_0000h	41.3.41/2096
E0	Tx Buffer Transmission Interrupt Enable Register (M_CAN_TXBTIE)	32	R/W	0000_0000h	41.3.42/2097
E4	Tx Buffer Cancellation Finished Interrupt Enable Register (M_CAN_TXBCIE)	32	R/W	0000_0000h	41.3.43/2097
F0	Tx Event FIFO Configuration Register (M_CAN_TXEFC)	32	R/W	0000_0000h	41.3.44/2098
F4	Tx Event FIFO Status Register (M_CAN_TXEFS)	32	R	0000_0000h	41.3.45/2099
F8	Tx Event FIFO Acknowledge Register (M_CAN_TXEFA)	32	R/W	0000_0000h	41.3.46/2100

41.3.1 Core Release Register (M_CAN_CREL)

The following table shows example field values for this register and explains how they encode a particular M_CAN core release.

REL	STEP	SUBSTEP	YEAR	MON	DAY	Release
0	2	0	9	3	26	Revision 0.2.0, date 2009/03/26

Memory Map and Register Description

Address: 0h base + 0h offset = 0h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	REL				STEP				SUBSTEP				YEAR				MON								DAY							
W																																
Reset	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*

* Notes:

- The coding of revisions depends on the module version used in the device.x = Undefined at reset.

M_CAN_CREL field descriptions

Field	Description
0–3 REL	Core Release One digit, BCD-coded.
4–7 STEP	Step of Core Release One digit, BCD-coded.
8–11 SUBSTEP	Sub-step of Core Release One digit, BCD-coded.
12–15 YEAR	Time Stamp Year One digit, BCD-coded. This field is set by generic parameter on M_CAN synthesis.
16–23 MON	Time Stamp Month Two digits, BCD-coded. This field is set by generic parameter on M_CAN synthesis.
24–31 DAY	Time Stamp Day Two digits, BCD-coded. This field is set by generic parameter on M_CAN synthesis.

41.3.2 Endian Register (M_CAN_ENDN)

Address: 0h base + 4h offset = 4h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	ETV																															
W																																
Reset	1	0	0	0	0	1	1	1	0	1	1	0	0	1	0	1	0	1	0	0	0	1	1	0	0	1	0	0	0	0	0	1

M_CAN_ENDN field descriptions

Field	Description
0–31 ETV	Endianness Test Value The endianness test value is 0x87654321.

41.3.3 Data Bit Timing and Prescaler Register (M_CAN_DBTP)

This register is only writable if bits CCCR.CCE and CCCR.INIT are set. The CAN bit time may be programmed in the range of 4 to 49 time quanta. The CAN time quantum may be programmed in the range of 1 to 32 m_can_cclk periods. $t_q = (DBRP + 1) \text{ mtq}$.

DTSEG1 is the sum of Prop_Seg and Phase_Seg1. DTSEG2 is Phase_Seg2.

Therefore the length of the bit time is (programmed values) $[DTSEG1 + DTSEG2 + 3] t_q$ or (functional values) $[Sync\_Seg + Prop\_Seg + Phase\_Seg1 + Phase\_Seg2] t_q$.

The Information Processing Time (IPT) is zero, meaning the data for the next bit is available at the first clock edge after the sample point.

NOTE

With a M_CAN clock of 8 MHz, the reset value of 0x00000A33 configures the M_CAN for a data phase bit rate of 500 kBit/s.

NOTE

The bit rate configured for the CAN FD data phase via DBTP must be higher or equal to the bit rate configured for the arbitration phase via NBTP.

Address: 0h base + Ch offset = Ch

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0							TDC	0		DBRP					
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0			DTSEG1					DTSEG2				DSJW			
W																
Reset	0	0	0	0	1	0	1	0	0	0	1	1	0	0	1	1

M_CAN_DBTP field descriptions

Field	Description
0–7 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
8 TDC	Transceiver Delay Compensation NOTE: This field has Protected Write status. 0 Transceiver Delay Compensation disabled 1 Transceiver Delay Compensation enabled

Table continues on the next page...

M_CAN_DBTP field descriptions (continued)

Field	Description
9–10 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
11–15 DBRP	Data Bit Rate Prescaler NOTE: This field has Protected Write status. 0x0-0x1F The value by which the oscillator frequency is divided for generating the bit time quanta. The bit time is built up from a multiple of this quanta. Valid values for the Bit Rate Prescaler are 0 to 31. The actual interpretation by the hardware of this value is such that one more than the value programmed here is used.
16–18 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
19–23 DTSEG1	Data time segment before sample point NOTE: This field has Protected Write status. 0x0-0x1F Valid values are 0 to 31. The actual interpretation by the hardware of this value is such that one more than the programmed value is used.
24–27 DTSEG2	Data time segment after sample point NOTE: This field has Protected Write status. 0x0-0xF Valid values are 0 to 15. The actual interpretation by the hardware of this value is such that one more than the programmed value is used.
28–31 DSJW	Data (Re)Synchronization Jump Width NOTE: This field has Protected Write status. 0x0-0xF Valid values are 0 to 15. The actual interpretation by the hardware of this value is such that one more than the value programmed here is used.

41.3.4 Test Register (M_CAN_TEST)

Write access to the Test Register has to be enabled by setting bit CCCR[TEST] to 1. All Test Register functions are set to their reset values when bit CCCR[TEST] is reset.

Loop Back Mode and software control of pin M_CAN_Tx are hardware test modes. Programming of TX other than 00 may disturb the message transfer on the CAN bus.

Address: 0h base + 10h offset = 10h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0								RX	TX		LBCK	0			
W																
Reset	0	0	0	0	0	0	0	0	u*	0	0	0	0	0	0	0

* Notes:

- u = Unaffected by reset.

M_CAN_TEST field descriptions

Field	Description
0–23 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
24 RX	Receive Pin Monitors the actual value of pin M_CAN Rx 0 The CAN bus is dominant (M_CAN Rx = 0) 1 The CAN bus is recessive (M_CAN Rx = 1)
25–26 TX	Control of Transmit Pin NOTE: This field has Protected Write status. 00 Reset value, M_CAN Tx controlled by the CAN Core, updated at the end of the CAN bit time 01 Sample Point can be monitored at pin M_CAN Tx 10 Dominant (0) level at pin M_CAN Tx 11 Recessive (1) at pin M_CAN Tx
27 LBCK	Loopback mode NOTE: This field has Protected Write status.

Table continues on the next page...

M_CAN_TEST field descriptions (continued)

Field	Description
	0 Reset value, Loopback mode is disabled
	1 Loopback mode is enabled (see Test Modes)
28–31 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

41.3.5 RAM Watchdog Register (M_CAN_RWD)

The RAM Watchdog monitors the READY output of the Message RAM. A Message RAM access via the M_CAN's Generic Master Interface starts the Message RAM Watchdog Counter with the value configured by RWD[WDC]. The counter is reloaded with RWD[WDC] when the Message RAM signals successful completion by activating its READY output. In case there is no response from the Message RAM until the counter has counted down to zero, the counter stops and interrupt flag IR[WDI] is set. The RAM Watchdog Counter is clocked by the Host clock.

Address: 0h base + 14h offset = 14h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0																WDV								WDC							
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

M_CAN_RWD field descriptions

Field	Description
0–15 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
16–23 WDV	Watchdog Value Actual Message RAM Watchdog Counter Value.
24–31 WDC	Watchdog Configuration Start value of the Message RAM Watchdog Counter. With the reset value of 00 the counter is disabled. NOTE: This field has Protected Write status.

41.3.6 CC Control Register (M_CAN_CCCR)

For details about setting and resetting of single bits see [Software Initialization](#).

Address: 0h base + 18h offset = 18h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R					0								CSA			
W	NISO	TXP	EFBI	PHDX			BRSE	FDOE	TEST	DAR	MON	CSR		ASM	CCE	INIT
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1

M_CAN_CCCR field descriptions

Field	Description
0–15 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
16 NISO	Non ISO Operation If this bit is set, the M_CAN uses the CAN FD frame format as specified by the Bosch CAN FD Specification V1.0. NOTE: This field has Protected Write status. 0 CAN FD frame format according to ISO11898-1 1 CAN FD frame format according to Bosch CAN FD Specification V1.0
17 TXP	Transmit Pause If this bit is set, the M_CAN pauses for two CAN bit times before starting the next transmission after itself has successfully transmitted a frame (see Tx Handling). NOTE: This field has Protected Write status.

Table continues on the next page...

M_CAN_CCCR field descriptions (continued)

Field	Description
	0 Transmit pause disabled 1 Transmit pause enabled
18 EFBI	Edge Filtering during Bus Integration NOTE: This field has Protected Write status. 0 Edge filtering disabled 1 Two consecutive dominant tq required to detect an edge for hard synchronization
19 PHDX	Protocol Exception Handling Disable NOTE: When protocol exception handling is disabled, the M_CAN will transmit an error frame when it detects a protocol exception condition. This field has Protected Write status. 0 Protocol exception handling enabled 1 Protocol exception handling disabled
20–21 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
22 BRSE	Bit Rate Switch Enable NOTE: When CAN FD operation is disabled FDOE = 0, BRSE is not evaluated. This field has Protected Write status. 0 Bit rate switching for transmissions disabled 1 Bit rate switching for transmissions enabled
23 FDOE	FD Operation Enable NOTE: This field has Protected Write status. 0 FD operation disabled 1 FD operation enabled
24 TEST	Test Mode Enable NOTE: This field has Protected Write status. 0 Normal operation, register TEST holds reset values 1 Test Mode, write access to register TEST enabled
25 DAR	Disable Automatic Retransmission NOTE: This field has Protected Write status. 0 Automatic retransmission of messages not transmitted successfully enabled 1 Automatic retransmission disabled
26 MON	Bus Monitoring Mode

Table continues on the next page...

M_CAN_CCCR field descriptions (continued)

Field	Description
	Bit MON can only be set by the Host when both CCE and INIT are set to '1'. The bit can be reset by the Host at any time. NOTE: This field has Protected Write status. 0 Bus Monitoring Mode is disabled 1 Bus Monitoring Mode is enabled
27 CSR	Clock Stop Request 0 No clock stop is requested 1 Clock stop requested. When clock stop is requested, first INIT and then CSA will be set after all pending transfer requests have been completed and the CAN bus reached idle.
28 CSA	Clock Stop Acknowledge 0 No clock stop acknowledged 1 M_CAN may be set in power down by stopping M_CAN input clocks.
29 ASM	Restricted Operation Mode Bit ASM can only be set by the Host when both CCE and INIT are set to '1'. The bit can be reset by the Host at any time. For a description of the Restricted Operation Mode see Restricted Operation Mode. 0 Normal CAN operation 1 Restricted Operation Mode active
30 CCE	Configuration Change Enable NOTE: This field has Protected Write status. 0 The CPU has no write access to the protected configuration registers. 1 The CPU has write access to the protected configuration registers (while CCCR[INIT] = 1)
31 INIT	Initialization NOTE: Due to the synchronization mechanism between the two clock domains, there may be a delay until the value written to INIT can be read back. Therefore the programmer has to assure that the previous value written to INIT has been accepted by reading INIT before setting INIT to a new value. 0 Normal Operation 1 Initialization is started

41.3.7 Nominal Bit Timing and Prescaler Register (M_CAN_NBTP)

This register is only writable if its CCCR[CCE] and CCCR[INIT] are set. The CAN bit time may be programmed in the range of 4 to 385 time quanta. The CAN time quantum may be programmed in the range of 1 to 512 m_can_cclk periods. $tq = (NBRP + 1) mtq$.

NTSEG1 is the sum of Prop_Seg and Phase_Seg1. NTSEG2 is Phase_Seg2.

Memory Map and Register Description

Therefore the length of the bit time is (programmed values) $[NTSEG1 + NTSEG2 + 3] t_q$ or (functional values) $[Sync\_Seg + Prop\_Seg + Phase\_Seg1 + Phase\_Seg2] t_q$.

The Information Processing Time (IPT) is zero, meaning the data for the next bit is available at the first clock edge after the sample point.

NOTE

With a CAN clock of 8 MHz, the reset value of 0x06000A03 configures the M_CAN for a bit rate of 500 kBit/s.

Address: 0h base + 1Ch offset = 1Ch

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	NSJW							NBRP								
W																
Reset	0	0	0	0	0	1	1	0	0	0	0	0	0	0	0	0
Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	NTSEG1							0	NTSEG2							
W																
Reset	0	0	0	0	1	0	1	0	0	0	0	0	0	0	1	1

M_CAN_NBTP field descriptions

Field	Description
0–6 NSJW	Nominal (Re) Synchronization Jump Width NOTE: This field has Protected Write status. 0x00-0x7F Valid values are 0 to 127. The actual interpretation by the hardware of this value is such that one more than the value programmed here is used.
7–15 NBRP	Nominal Baud Rate Prescaler NOTE: This field has Protected Write status. 0x000-0x1FF The value by which the oscillator frequency is divided for generating the bit time quanta. The bit time is built up from a multiple of this quanta. Valid values for the Bit Rate Prescaler are 0 to 511. The actual interpretation by the hardware of this value is such that one more than the value programmed here is used.
16–23 NTSEG1	Nominal Time segment before sample point NOTE: This field has Protected Write status. 0x01-0xFF Valid values are 1 to 255. The actual interpretation by the hardware of this value is such that one more than the programmed value is used.
24 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
25–31 NTSEG2	Nominal Time segment after sample point NOTE: This field has Protected Write status.

Table continues on the next page...

M_CAN_NBTP field descriptions (continued)

Field	Description
0x00-0x7F	Valid values are 0 to 127. The actual interpretation by the hardware of this value is such that one more than the value programmed here is used.

41.3.8 Timestamp Counter Configuration Register (M_CAN_TSCC)

For description of the Timestamp Counter see [Timestamp Generation](#)

Address: 0h base + 20h offset = 20h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0												TCP				0													TSS		
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	

M_CAN_TSCC field descriptions

Field	Description
0–11 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
12–15 TCP	Timestamp Counter Prescaler NOTE: This field has Protected Write status. NOTE: With CAN FD, timestamp generation is not supported. 0x0-0xF Configures the timestamp and timeout counters time unit in multiples of CAN bit times [1...16]. The actual interpretation by the hardware of this value is such that one more than the value programmed here is used.
16–29 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
30–31 TSS	Timestamp Select NOTE: This field has Protected Write status. 00 Timestamp counter value always 0x0000 01 Timestamp counter value incremented according to TCP 10 Reserved 11 Same as 00

41.3.9 Timestamp Counter Value Register (M_CAN_TSCV)

Address: 0h base + 24h offset = 24h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0																TSC															
W																	w1c															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	

M_CAN_TSCV field descriptions

Field	Description
0–15 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
16–31 TSC	Timestamp Counter The internal Timestamp Counter value is captured on start of frame (both Rx and Tx). When TSCC[TSS] = 01, the Timestamp Counter is incremented in multiples of CAN bit times [1...16] depending on the configuration of TSCC[TCP]. A wrap around sets interrupt flag IR[TSW]. Write access resets the counter to zero. NOTE: A "wrap around" is a change of the Timestamp Counter value from non-zero to zero not caused by write access to TSCV.

41.3.10 Timeout Counter Configuration Register (M_CAN_TOCC)

See [Timeout Counter](#) for description of the Timeout Counter or for use of timeout function with CAN FD.

Address: 0h base + 28h offset = 28h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	
R	TOP																
W																	
Reset	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	
Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	
R	0												TOS				ETOC
W																	
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	

M_CAN_TOCC field descriptions

Field	Description
0–15 TOP	Timeout Period Start value of the Timeout Counter (down-counter). Configures the Timeout Period. NOTE: This field has Protected Write status.
16–28 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
29–30 TOS	Timeout Select When operating in Continuous mode, a write to TOCV presets the counter to the value configured by TOCC[TOP] and continues down-counting. When the Timeout Counter is controlled by one of the FIFOs, an empty FIFO presets the counter to the value configured by TOCC[TOP]. Down-counting is started when the first FIFO element is stored. NOTE: This field has Protected Write status. 00 Continuous operation 01 Timeout controlled by Tx Event FIFO 10 Timeout controlled by Rx FIFO 0 11 Timeout controlled by Rx FIFO 1
31 ETOC	Enable Timeout Counte NOTE: This field has Protected Write status. 0 Timeout Counter disabled 1 Timeout Counter enabled

41.3.11 Timeout Counter Value Register (M_CAN_TOCV)

Address: 0h base + 2Ch offset = 2Ch

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0																TOC															
W																	w1c															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	

M_CAN_TOCV field descriptions

Field	Description
0–15 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
16–31 TOC	Timeout Counter The Timeout Counter is decremented in multiples of CAN bit times [1...16] depending on the configuration of TSSC[TCP]. When decremented to zero, interrupt flag IR[TOO] is set and the Timeout Counter is stopped. Start and reset/restart conditions are configured via TOCC[TOS]

41.3.12 Error Counter Register (M_CAN_ECR)

NOTE

When CCCR[ASM] is set, the CAN protocol controller does not increment TEC and REC when a CAN protocol error is detected, but CEL is still incremented.

Address: 0h base + 40h offset = 40h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	
R	0								CEL								
W									CEL								
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	
R	RP	REC							TEC								
W																	
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	

M_CAN_ECR field descriptions

Field	Description
0–7 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
8–15 CEL	CAN Error Logging The counter is incremented each time when a CAN protocol error causes the Transmit Error Counter or the Receive Error Counter to be incremented. It is reset by read access to CEL. The counter stops at 0xFF; the next increment of TEC or REC sets interrupt flag IR[ELO].
16 RP	Receive Error Passive 0 The Receive Error Counter is below the error passive level of 128 1 The Receive Error Counter has reached the error passive level of 128
17–23 REC	Receive Error Counter Actual state of the Receive Error Counter, values between 0 and 127.
24–31 TEC	Transmit Error Counter Actual state of the Transmit Error Counter, values between 0 and 255.

41.3.13 Protocol Status Register (M_CAN_PSR)

NOTE

When a frame in CAN FD format has reached the data phase with BRS flag set, the next CAN event (error or valid frame) will be shown in DLEC instead of LEC. An error in a fixed stuff bit of a CAN FD CRC sequence will be shown as a Form Error, not Stuff Error.

NOTE

The Bus_Off recovery sequence (see CAN Specification Rev. 2.0 or ISO11898-1) cannot be shortened by setting or resetting CCCR[INIT].

1. If the device goes Bus_Off, it sets CCCR[INIT] and PSR[BO], stopping all bus activities.
2. After the user clears CCCR[INIT], the user must then wait for 129 occurrences of Bus Idle (129 * 11 consecutive recessive bits) before normal CAN operation can resume.
 - During the waiting period, each time the device monitors a sequence of 11 recessive bits, a Bit0Error code (101b) is written to PSR[LEC].
 - These code writes enable the user to check readily whether the CAN bus is stuck at dominant or continuously disturbed and to monitor the Bus_Off recovery sequence.
 - ECR[REC] is used to count these sequences.
3. After the observation of the 129 occurrences of Bus Idle, the device clears PSR[BO].

Address: 0h base + 44h offset = 44h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0								TDCV							
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	PXE	RFDF	RBRS	RESI	DLEC			BO	EW	EP	ACT		LEC		
W																
Reset	0	0	0	0	0	1	1	1	0	0	0	0	0	1	1	1

M_CAN_PSR field descriptions

Field	Description
0–8 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
9–15 TDCV	Transmitter Delay Compensation Value 0x00-0x7F Position of the secondary sample point, defined by the sum of the measured delay from M_CAN Tx to M_CAN Rx and TDCR[TDCO]. The SSP position is, in the data phase, the number of mtq between the start of the transmitted bit and the secondary sample point. Valid values are 0 to 127 mtq.
16 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
17 PXE	Protocol Exception Event NOTE: This field is reset by a read operation. 0 No protocol exception event occurred since last read access 1 Protocol exception event occurred
18 RFDF	Received a CAN FD Message This bit is set independent of acceptance filtering. NOTE: This field is reset by a read operation. 0 Since this bit was reset by the CPU, no CAN FD message has been received 1 Message in CAN FD format with FDF flag set has been received
19 RBRS	BRS flag of last received CAN FD Message This bit is set together with RFDF, independent of acceptance filtering. NOTE: This field is reset by a read operation. 0 Last received CAN FD message did not have its BRS flag set 1 Last received CAN FD message had its BRS flag set
20 RESI	ESI flag of last received CAN FD Message This bit is set together with RFDF, independent of acceptance filtering. NOTE: This field is reset by a read operation. 0 Last received CAN FD message did not have its ESI flag set 1 Last received CAN FD message had its ESI flag set
21–23 DLEC	Data Phase Last Error Code Type of last error that occurred in the data phase of a CAN FD format frame with its BRS flag set. Coding is the same as for LEC. This field will be cleared to zero when a CAN FD format frame with its BRS flag set has been transferred (reception or transmission) without error. NOTE: This field is set by a read operation.
24 BO	Bus_Off Status 0 The M_CAN is not Bus_Off 1 The M_CAN is in Bus_Off state

Table continues on the next page...

M_CAN_PSR field descriptions (continued)

Field	Description
25 EW	Warning Status 0 Both error counters are below the Error_Warning limit of 96 1 At least one of error counter has reached the Error_Warning limit of 96
26 EP	Error Passive 0 The M_CAN is in the Error_Active state. It normally takes part in bus communication and sends an active error flag when an error has been detected 1 The M_CAN is in the Error_Passive state
27–28 ACT	Activity Monitors the module's CAN communication state. NOTE: ACT is set to 00 by a Protocol Exception Event. 00 Synchronizing - node is synchronizing on CAN communication 01 Idle - node is neither receiver nor transmitter 10 Receiver - node is operating as receiver 11 Transmitter - node is operating as transmitter
29–31 LEC	Last Error Code The LEC indicates the type of the last error to occur on the CAN bus. This field will be cleared to 0 when a message has been transferred (reception or transmission) without error. NOTE: This field is set by a read operation. 000 Error: No error occurred since LEC has been reset by successful reception or transmission 001 Stuff Error: More than 5 equal bits in a sequence have occurred in a part of a received message where this is not allowed 010 Form Error: A fixed format part of a received frame has the wrong format 011 AckError: The message transmitted by the M_CAN was not acknowledged by another node. 100 Bit1Error: During the transmission of a message (with the exception of the arbitration field), the device wanted to send a recessive level (bit of logical value 1), but the monitored bus value was dominant. 101 Bit0Error: During the transmission of a message (or acknowledge bit, or active error flag, or overload flag), the device wanted to send a dominant level (data or identifier bit logical value '0'), but the monitored bus value was recessive. During Bus_Off recovery this status is set each time a sequence of 11 recessive bits has been monitored. This enables the CPU to monitor the proceeding of the Bus_Off recovery sequence (indicating the bus is not stuck at dominant or continuously disturbed). 110 CRCError: The CRC check sum of a received message was incorrect. The CRC of an incoming message does not match with the CRC calculated from the received data. 111 NoChange: Any read access to the Protocol Status Register re-initializes the LEC to 7. When the LEC shows the value 7, no CAN bus event was detected since the last CPU read access to the Protocol Status Register.

41.3.14 Transmitter Delay Compensation Register (M_CAN_TDCR)

Address: 0h base + 48h offset = 48h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	TDCO							0	TDCF						
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

M_CAN_TDCR field descriptions

Field	Description
0–16 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
17–23 TDCO	Transmitter Delay Compensation Offset. NOTE This field has Protected Write status. 0x00-0x7F Offset value defining the distance between the measured delay from M_CAN Tx to M_CAN Rx and the secondary sample point. Valid values are 0 to 127 mtq.
24 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
25–31 TDCF	Transmitter Delay Compensation Filter Window Length. NOTE This field has Protected Write status. 0x00-0x7F Defines the minimum value for the SSP position, dominant edges on M_CAN Rx that would result in an earlier SSP position are ignored for transmitter delay measurement. The feature is enabled when TDCF is configured to a value greater than TDCO. Valid values are 0 to 127 mtq.

41.3.15 Interrupt Register (M_CAN_IR)

The flags are set when one of the listed conditions is detected (edge-sensitive). The flags remain set until the Host clears them. Aflag is cleared by writing a 1 to the corresponding bit position. Writing a 0 has no effect. Ahard reset will clear the register. The configuration of IE controls whether an interrupt is generated. The configuration of ILS controls on which interrupt line an interrupt is signalled.

Address: 0h base + 50h offset = 50h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0															
W			ARA	PED	PEA	WDI	BO	EW	EP	ELO	BEU	BEC	DRX	TOO	MRAF	TSW
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R																
W	TEFL	TEFF	TEFW	TEFN	TFE	TCF	TC	HPM	RF1L	RF1F	RF1W	RF1N	RF0L	RF0F	RF0W	RF0N
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

M_CAN_IR field descriptions

Field	Description
0–1 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
2 ARA	Access to Reserved Address 0 No access to reserved address occurred 1 Access to reserved address occurred
3 PED	Protocol Error in Data Phase 0 No protocol error in data phase 1 Protocol error in data phase detected (PSR[DLEC] not equal to 0,7)
4 PEA	Protocol Error in Arbitration Phase 0 No protocol error in arbitration phase 1 Protocol error in arbitration phase detected (PSR[LEC] not equal to 0,7)
5 WDI	Watchdog Interrupt 0 No Message RAM Watchdog event occurred 1 Message RAM Watchdog event due to missing READY
6 BO	Bus_Off Status

Table continues on the next page...

M_CAN_IR field descriptions (continued)

Field	Description
	0 Bus_Off status unchanged 1 Bus_Off status changed
7 EW	Warning Status 0 Error_Warning status unchanged 1 Error_Warning status changed
8 EP	Error Passive 0 Error_Passive status unchanged 1 Error_Passive status changed
9 ELO	Error Logging Overflow 0 CAN Error Logging Counter did not overflow 1 Overflow of CAN Error Logging Counter occurred
10 BEU	Bit Error Uncorrected Message RAM bit error detected, uncorrected. Controlled by an external ECC logic attached to the Message RAM. An uncorrected Message RAM bit error sets CCCR[INIT] to 1. This is done to avoid transmission of corrupted data. 0 No bit error detected when reading from Message RAM 1 Bit error detected, uncorrected (e.g. parity logic)
11 BEC	Bit Error Corrected Message RAM bit error detected and corrected. Controlled by an external parity ECC logic attached to the Message RAM. 0 No bit error detected when reading from Message RAM 1 Bit error detected and corrected (e.g. ECC)
12 DRX	Message stored to Dedicated Rx Buffer The flag is set whenever a received message has been stored into a dedicated Rx Buffer. 0 No Rx Buffer updated 1 At least one received message stored into a Rx Buffer
13 TOO	Timeout Occurred 0 No timeout 1 Timeout reached
14 MRAF	Message RAM Access Failure The flag is set, when the Rx Handler - has not completed acceptance filtering or storage of an accepted message until the arbitration field of the following message has been received. In this case acceptance filtering or message storage is aborted and the Rx Handler starts processing of the following message. - was not able to write a message to the Message RAM. In this case message storage is aborted. In both cases the FIFO put index is not updated resp. the New Data flag for a dedicated Rx Buffer is not set, a partly stored message is overwritten when the next message is stored to this location.

Table continues on the next page...

M_CAN_IR field descriptions (continued)

Field	Description
	The flag is also set when the Tx Handler was not able to read a message from the Message RAM in time. In this case message transmission is aborted. In case of a Tx Handler access failure the M_CAN is switched into Restricted Operation Mode (see Restricted Operation Mode). To leave Restricted Operation Mode, the CPU has to reset CCCR[ASM]. 0 No Message RAM access failure occurred 1 Message RAM access failure occurred
15 TSW	Timestamp Wraparound 0 No timestamp counter wrap-around 1 Timestamp counter wrapped around
16 TEFL	Tx Event FIFO Element Lost 0 No Tx Event FIFO element lost 1 Tx Event FIFO element lost, also set after write attempt to Tx Event FIFO of size zero
17 TEFF	Tx Event FIFO Full 0 Tx Event FIFO not full 1 Tx Event FIFO full
18 TEFW	Tx Event FIFO Watermark Reached 0 Tx Event FIFO fill level below watermark 1 Tx Event FIFO fill level reached watermark
19 TEFN	Tx Event FIFO New Entry 0 Tx Event FIFO unchanged 1 Tx Handler wrote Tx Event FIFO element
20 TFE	Tx FIFO Empty 0 Tx FIFO non-empty 1 Tx FIFO empty
21 TCF	Transmission Cancellation Finished 0 No transmission cancellation finished 1 Transmission cancellation finished
22 TC	Transmission Completed 0 Transmission not completed 1 Transmission completed
23 HPM	High Priority Message 0 No high priority message received 1 High priority message received
24 RF1L	Rx FIFO 1 Message Lost 0 No Rx FIFO 1 message lost 1 Rx FIFO 1 message lost, also set after write attempt to Rx FIFO 1 of size zero
25 RF1F	Rx FIFO 1 Full

Table continues on the next page...

M_CAN_IR field descriptions (continued)

Field	Description
	0 Rx FIFO 1 not full 1 Rx FIFO 1 full
26 RF1W	Rx FIFO 1 Watermark Reached 0 Rx FIFO 1 fill level below watermark 1 Rx FIFO 1 fill level reached watermark
27 RF1N	Rx FIFO 1 New Message 0 No new message written to Rx FIFO 1 1 New message written to Rx FIFO 1
28 RF0L	Rx FIFO 0 Message Lost 0 No Rx FIFO 0 message lost 1 Rx FIFO 0 message lost, also set after write attempt to Rx FIFO 0 of size zero
29 RF0F	Rx FIFO 0 Full 0 Rx FIFO 0 not full 1 Rx FIFO 0 full
30 RF0W	Rx FIFO 0 Watermark Reached 0 Rx FIFO 0 fill level below watermark 1 Rx FIFO 0 fill level reached watermark
31 RF0N	Rx FIFO 0 New Message 0 No new message written to Rx FIFO 0 1 New message written to Rx FIFO 0

41.3.16 Interrupt Enable Register (M_CAN_IE)

The settings in the Interrupt Enable register determine which status changes in the Interrupt Register will be signaled on an interrupt line.

Address: 0h base + 54h offset = 54h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0		ARAE	PEDE	PEAE	WDIE	BOE	EWE	EPE	ELOE	BEUE	BECE	DRXE	TOOE	MRAFE	TSWE
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	TEFLE	TEFFE	TEFWE	TEFNE	TFEE	TCFE	TCE	HPME	RF1LE	RF1FE	RF1WE	RF1NE	RF0LE	RF0FE	RF0WE	RF0NE
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

M_CAN_IE field descriptions

Field	Description
0–1 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
2 ARAE	Access to Reserved Address Enable
3 PEDE	Protocol Error in Data Phase Enable
4 PEAE	Protocol Error in Arbitration Phase Enable
5 WDIE	Watchdog Interrupt Enable 0 Interrupt disabled 1 Interrupt enabled
6 BOE	Bus_Off Status Interrupt Enable 0 Interrupt disabled 1 Interrupt enabled
7 EWE	Warning Status Interrupt Enable 0 Interrupt disabled 1 Interrupt enabled
8 EPE	Error Passive Interrupt Enable 0 Interrupt disabled 1 Interrupt enabled
9 ELOE	Error Logging Overflow Interrupt Enable 0 Interrupt disabled 1 Interrupt enabled
10 BEUE	Bit Error Uncorrected Interrupt Enable 0 Interrupt disabled 1 Interrupt enabled
11 BECE	Bit Error Corrected Interrupt Enable 0 Interrupt disabled 1 Interrupt enabled
12 DRXE	Message stored to Dedicated Rx Buffer Interrupt Enable 0 Interrupt disabled 1 Interrupt enabled
13 TOOE	Timeout Occurred Interrupt Enable 0 Interrupt disabled 1 Interrupt enabled
14 MRAFE	Message RAM Access Failure Interrupt Enable 0 Interrupt disabled 1 Interrupt enabled

Table continues on the next page...

M_CAN_IE field descriptions (continued)

Field	Description
15 TSWE	Timestamp Wraparound Interrupt Enable 0 Interrupt disabled 1 Interrupt enabled
16 TEFLE	Tx Event FIFO Element Lost Interrupt Enable 0 Interrupt disabled 1 Interrupt enabled
17 TEFFE	Tx Event FIFO Full Interrupt Enable 0 Interrupt disabled 1 Interrupt enabled
18 TEFWE	Tx Event FIFO Watermark Reached Interrupt Enable 0 Interrupt disabled 1 Interrupt enabled
19 TEFNE	Tx Event FIFO New Entry Interrupt Enable 0 Interrupt disabled 1 Interrupt enabled
20 TFEE	Tx FIFO Empty Interrupt Enable 0 Interrupt disabled 1 Interrupt enabled
21 TCFE	Transmission Cancellation Finished Interrupt Enable 0 Interrupt disabled 1 Interrupt enabled
22 TCE	Transmission Completed Interrupt Enable 0 Interrupt disabled 1 Interrupt enabled
23 HPME	High Priority Message Interrupt Enable 0 Interrupt disabled 1 Interrupt enabled
24 RF1LE	Rx FIFO 1 Message Lost Interrupt Enable 0 Interrupt disabled 1 Interrupt enabled
25 RF1FE	Rx FIFO 1 Full Interrupt Enable 0 Interrupt disabled 1 Interrupt enabled
26 RF1WE	Rx FIFO 1 Watermark Reached Interrupt Enable 0 Interrupt disabled 1 Interrupt enabled

Table continues on the next page...

M_CAN_IE field descriptions (continued)

Field	Description
27 RF1NE	Rx FIFO 1 New Message Interrupt Enable 0 Interrupt disabled 1 Interrupt enabled
28 RF0LE	Rx FIFO 0 Message Lost Interrupt Enable 0 Interrupt disabled 1 Interrupt enabled
29 RF0FE	Rx FIFO 0 Full Interrupt Enable 0 Interrupt disabled 1 Interrupt enabled
30 RF0WE	Rx FIFO 0 Watermark Reached Interrupt Enable 0 Interrupt disabled 1 Interrupt enabled
31 RF0NE	Rx FIFO 0 New Message Interrupt Enable 0 Interrupt disabled 1 Interrupt enabled

41.3.17 Interrupt Line Select Register (M_CAN_ILS)

The Interrupt Line Select register assigns an interrupt generated by a specific interrupt flag from the Interrupt Register to one of the two module interrupt lines. For interrupt generation the respective interrupt line has to be enabled via ILE[EINT0] and ILE[EINT1]

0 = Interrupt assigned to interrupt line 0

1 = Interrupt assigned to interrupt line 1

Address: 0h base + 58h offset = 58h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0															
W			ARAL	PEDL	PEAL	WDIL	BOL	EWL	EPL	ELOL	BEUL	BECL	DRXL	TOOL	MRAFL	TSWL
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Memory Map and Register Description

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	TEFL	TEFL	TEFWL	TEFNL	TFEL	TCFL	TCL	HPML	RF1LL	RF1FL	RF1WL	RF1NL	RF0LL	RF0FL	RF0WL	RF0NL
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

M_CAN_ILS field descriptions

Field	Description
0–1 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
2 ARAL	Access to Reserved Address Line 0 Interrupt assigned to M_CAN interrupt line 0 1 Interrupt assigned to M_CAN interrupt line 1
3 PEDL	Protocol Error in Data Phase Line 0 Interrupt assigned to M_CAN interrupt line 0 1 Interrupt assigned to M_CAN interrupt line 1
4 PEAL	Protocol Error in Arbitration Phase Line 0 Interrupt assigned to M_CAN interrupt line 0 1 Interrupt assigned to M_CAN interrupt line 1
5 WDIL	Watchdog Interrupt Line 0 Interrupt assigned to M_CAN interrupt line 0 1 Interrupt assigned to M_CAN interrupt line 1
6 BOL	Bus_Off Status Interrupt Line 0 Interrupt assigned to M_CAN interrupt line 0 1 Interrupt assigned to M_CAN interrupt line 1
7 EWL	Warning Status Interrupt Line 0 Interrupt assigned to M_CAN interrupt line 0 1 Interrupt assigned to M_CAN interrupt line 1
8 EPL	Error Passive Interrupt Line 0 Interrupt assigned to M_CAN interrupt line 0 1 Interrupt assigned to M_CAN interrupt line 1
9 ELOL	Error Logging Overflow Interrupt Line 0 Interrupt assigned to M_CAN interrupt line 0 1 Interrupt assigned to M_CAN interrupt line 1
10 BEUL	Bit Error Uncorrected Interrupt Line 0 Interrupt assigned to M_CAN interrupt line 0 1 Interrupt assigned to M_CAN interrupt line 1
11 BECL	Bit Error Corrected Interrupt Line

Table continues on the next page...

M_CAN_ILS field descriptions (continued)

Field	Description
	0 Interrupt assigned to M_CAN interrupt line 0 1 Interrupt assigned to M_CAN interrupt line 1
12 DRXL	Message stored to Dedicated Rx Buffer Interrupt Line 0 Interrupt assigned to M_CAN interrupt line 0 1 Interrupt assigned to M_CAN interrupt line 1
13 TOOL	Timeout Occurred Interrupt Line 0 Interrupt assigned to M_CAN interrupt line 0 1 Interrupt assigned to M_CAN interrupt line 1
14 MRAFL	Message RAM Access Failure Interrupt Line 0 Interrupt assigned to M_CAN interrupt line 0 1 Interrupt assigned to M_CAN interrupt line 1
15 TSWL	Timestamp Wraparound Interrupt Line 0 Interrupt assigned to M_CAN interrupt line 0 1 Interrupt assigned to M_CAN interrupt line 1
16 TEFLL	Tx Event FIFO Element Lost Interrupt Line 0 Interrupt assigned to M_CAN interrupt line 0 1 Interrupt assigned to M_CAN interrupt line 1
17 TEFFL	Tx Event FIFO Full Interrupt Line 0 Interrupt assigned to M_CAN interrupt line 0 1 Interrupt assigned to M_CAN interrupt line 1
18 TEFWL	Tx Event FIFO Watermark Reached Interrupt Line 0 Interrupt assigned to M_CAN interrupt line 0 1 Interrupt assigned to M_CAN interrupt line 1
19 TEFNL	Tx Event FIFO New Entry Interrupt Line 0 Interrupt assigned to M_CAN interrupt line 0 1 Interrupt assigned to M_CAN interrupt line 1
20 TFEL	Tx FIFO Empty Interrupt Line 0 Interrupt assigned to M_CAN interrupt line 0 1 Interrupt assigned to M_CAN interrupt line 1
21 TCFL	Transmission Cancellation Finished Interrupt Line 0 Interrupt assigned to M_CAN interrupt line 0 1 Interrupt assigned to M_CAN interrupt line 1
22 TCL	Transmission Completed Interrupt Line 0 Interrupt assigned to M_CAN interrupt line 0 1 Interrupt assigned to M_CAN interrupt line 1
23 HPML	High Priority Message Interrupt Line

Table continues on the next page...

M_CAN_ILS field descriptions (continued)

Field	Description
	0 Interrupt assigned to M_CAN interrupt line 0 1 Interrupt assigned to M_CAN interrupt line 1
24 RF1LL	Rx FIFO 1 Message Lost Interrupt Line 0 Interrupt assigned to M_CAN interrupt line 0 1 Interrupt assigned to M_CAN interrupt line 1
25 RF1FL	Rx FIFO 1 Full Interrupt Line 0 Interrupt assigned to M_CAN interrupt line 0 1 Interrupt assigned to M_CAN interrupt line 1
26 RF1WL	Rx FIFO 1 Watermark Reached Interrupt Line 0 Interrupt assigned to M_CAN interrupt line 0 1 Interrupt assigned to M_CAN interrupt line 1
27 RF1NL	Rx FIFO 1 New Message Interrupt Line 0 Interrupt assigned to M_CAN interrupt line 0 1 Interrupt assigned to M_CAN interrupt line 1
28 RF0LL	Rx FIFO 0 Message Lost Interrupt Line 0 Interrupt assigned to M_CAN interrupt line 0 1 Interrupt assigned to M_CAN interrupt line 1
29 RF0FL	Rx FIFO 0 Full Interrupt Line 0 Interrupt assigned to M_CAN interrupt line 0 1 Interrupt assigned to M_CAN interrupt line 1
30 RF0WL	Rx FIFO 0 Watermark Reached Interrupt Line 0 Interrupt assigned to M_CAN interrupt line 0 1 Interrupt assigned to M_CAN interrupt line 1
31 RF0NL	Rx FIFO 0 New Message Interrupt Line 0 Interrupt assigned to M_CAN interrupt line 0 1 Interrupt assigned to M_CAN interrupt line 1

41.3.18 Interrupt Line Enable Register (M_CAN_ILE)

Each of the two interrupt lines to the CPU can be enabled / disabled separately by programming bits EINT0 and EINT1.

Address: 0h base + 5Ch offset = 5Ch

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15															
R	0																														
W																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0															
Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31															
R	0																													EINT1	EINT0
W																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0															

M_CAN_ILE field descriptions

Field	Description
0–29 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
30 EINT1	Enable Interrupt Line 1 0 M_CAN interrupt line 1 disabled 1 M_CAN interrupt line 1 enabled
31 EINT0	Enable Interrupt Line 0 0 M_CAN interrupt line 0 disabled 1 M_CAN interrupt line 0 enabled

41.3.19 Global Filter Configuration Register (M_CAN_GFC)

Global settings for Message ID filtering. The Global Filter Configuration controls the filter path for standard and extended messages as described in the [Standard Message ID Filtering](#) and [Extended Message ID Filtering](#).

Address: 0h base + 80h offset = 80h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0										ANFS		ANFE		RRFS	RRFE
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

M_CAN_GFC field descriptions

Field	Description
0–25 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
26–27 ANFS	Accept Non-matching Frames Standard Defines how received messages with 11-bit IDs that do not match any element of the filter list are treated. NOTE: This field has Protected Write status. 00 Accept in Rx FIFO 0 01 Accept in Rx FIFO 1 10 Reject 11 Reject
28–29 ANFE	Accept Non-matching Frames Extended Defines how received messages with 29-bit IDs that do not match any element of the filter list are treated. NOTE: This field has Protected Write status. 00 Accept in Rx FIFO 0 01 Accept in Rx FIFO 1 10 Reject 11 Reject
30 RRFS	Reject Remote Frames Standard NOTE: This field has Protected Write status. 0 Filter remote frames with 11-bit standard IDs 1 Reject all remote frames with 11-bit standard IDs

Table continues on the next page...

M_CAN_GFC field descriptions (continued)

Field	Description
31 RRFE	Reject Remote Frames Extended NOTE: This field has Protected Write status. 0 Filter remote frames with 29-bit extended IDs 1 Reject all remote frames with 29-bit extended IDs

41.3.20 Standard ID Filter Configuration Register (M_CAN_SIDFC)

Settings for 11-bit standard Message ID filtering. The Standard ID Filter Configuration controls the filter path for the standard messages as described in [Standard Message ID Filtering](#).

Address: 0h base + 84h offset = 84h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0								LSS								FLSSA												0			
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	

M_CAN_SIDFC field descriptions

Field	Description
0–7 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
8–15 LSS	List Size Standard NOTE: This field has Protected Write status. 0 No standard Message ID filter 1–128 Number of standard Message ID filter elements >128 Values greater than 128 are interpreted as 128
16–29 FLSSA	Filter List Standard Start Address Start address of standard Message ID filter list (32-bit word address, see Message RAM). NOTE: This field has Protected Write status.
30–31 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

41.3.21 Extended ID Filter Configuration Register (M_CAN_XIDFC)

Settings for 29-bit extended Message ID filtering. The Extended ID Filter Configuration controls the filter path for the standard messages as described in [Extended Message ID Filtering](#).

Address: 0h base + 88h offset = 88h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R					0																											0
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

M_CAN_XIDFC field descriptions

Field	Description
0–8 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
9–15 LSE	List Size Extended NOTE: This field has Protected Write status. 0 No extended Message ID filter 1–64 Number of extended Message ID filter elements >64 Values greater than 64 are interpreted as 64
16–29 FLESA	Filter List Extended Start Address Start address of extended Message ID filter list (32-bit word address, see Message RAM). NOTE: This field has Protected Write status.
30–31 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

41.3.22 Extended ID and Mask Register (M_CAN_XIDAM)

Address: 0h base + 90h offset = 90h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0																															
W																																
Reset	0	0	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

M_CAN_XIDAM field descriptions

Field	Description
0–2 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
3–31 EIDM	Extended ID Mask For acceptance filtering of extended frames the Extended ID and Mask is ANDed with the Message ID of a received frame. Intended for masking of 29-bit IDs in SAE J1939. With the reset value of all bits set to one the mask is not active. NOTE: This field has Protected Write status.

41.3.23 High Priority Message Status Register (M_CAN_HPMS)

This register is updated every time a Message ID filter element configured to generate a priority event matches. This can be used to monitor the status of incoming high priority messages and to enable fast access to these messages.

Address: 0h base + 94h offset = 94h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	FLST	FIDX							MSI		BIDX					
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

M_CAN_HPMS field descriptions

Field	Description
0–15 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
16 FLST	Filter List Indicates the filter list of the matching filter element. 0 Standard Filter List 1 Extended Filter List
17–23 FIDX	Filter Index Index of matching filter element. Range is 0 to SIDFC[LSS] - 1 resp. XIDFC[LSE] - 1.
24–25 MSI	Message Storage Indicator

Table continues on the next page...

M_CAN_HPMS field descriptions (continued)

Field	Description
	00 No FIFO selected 01 FIFO message lost 10 Message stored in FIFO 0 11 Message stored in FIFO 1
26–31 BIDX	Buffer Index Index of Rx FIFO element to which the message was stored. Only valid when MSI[1] = 1.

41.3.24 New Data 1 Register (M_CAN_NDAT1)

Address: 0h base + 98h offset = 98h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R																																
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

M_CAN_NDAT1 field descriptions

Field	Description
0–31 ND1	New Data[0:31] The register holds the New Data flags of Rx Buffers 0 to 31. The flags are set when the respective Rx Buffer has been updated from a received frame. The flags remain set until the CPU clears them. A flag is cleared by writing a 1 to the corresponding bit position. Writing a 0 has no effect. A hard reset will clear the register. 0 Rx Buffer not updated 1 Rx Buffer updated from new message

41.3.25 New Data 2 Register (M_CAN_NDAT2)

Address: 0h base + 9Ch offset = 9Ch

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R																																
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

M_CAN_NDAT2 field descriptions

Field	Description
0–31 ND2	New Data[32:63] The register holds the New Data flags of Rx Buffers 32 to 63. The flags are set when the respective Rx Buffer has been updated from a received frame. The flags remain set until the CPU clears them. A flag is cleared by writing a 1 to the corresponding bit position. Writing a 0 has no effect. A hard reset will clear the register.

M_CAN_NDAT2 field descriptions (continued)

Field	Description
0	Rx Buffer not updated
1	Rx Buffer updated from new message

41.3.26 Rx FIFO 0 Configuration Register (M_CAN_RXF0C)

Address: 0h base + A0h offset = A0h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R									0							
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R																0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

M_CAN_RXF0C field descriptions

Field	Description
0 F0OM	FIFO 0 Operation Mode FIFO 0 can be operated in blocking or in overwrite mode (see Rx FIFOs). NOTE: This field has Protected Write status. 0 FIFO 0 blocking mode 1 FIFO 0 overwrite mode
1–7 F0WM	Rx FIFO 0 Watermark NOTE: This field has Protected Write status. 0 Watermark interrupt disabled 1–64 Level for Rx FIFO 0 watermark interrupt (IR[RF0W]) >64 Watermark interrupt disabled
8 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
9–15 F0S	Rx FIFO 0 Size The Rx FIFO 0 elements are indexed from 0 to F0S-1. NOTE: This field has Protected Write status.

Table continues on the next page...

M_CAN_RXF0C field descriptions (continued)

Field	Description
0	No Rx FIFO 0
1-64	Number of Rx FIFO 0 elements
>64	Values greater than 64 are interpreted as 64
16-29 F0SA	Rx FIFO 0 Start Address Start address of Rx FIFO 0 in Message RAM (32-bit word address, see Message RAM). NOTE: This field has Protected Write status.
30-31 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

41.3.27 Rx FIFO 0 Status Register (M_CAN_RXF0S)

Address: 0h base + A4h offset = A4h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0						RF0L	F0F	0	F0PI						
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31													
R	0	F0GI							0	F0FL																			
W																													
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0													

M_CAN_RXF0S field descriptions

Field	Description
0-5 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
6 RF0L	Rx FIFO 0 Message Lost This bit is a copy of interrupt flag IR[RF0L]. When IR[RF0L] is reset, this bit is also reset. NOTE: Overwriting the oldest message when RXF0C.FOOM = '1' will not set this flag. 0 No Rx FIFO 0 message lost 1 Rx FIFO 0 message lost, also set after write attempt to Rx FIFO 0 of size zero
7 F0F	Rx FIFO 0 Full 0 Rx FIFO 0 not full 1 Rx FIFO 0 full
8-9 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
10-15 F0PI	Rx FIFO 0 Put Index Rx FIFO 0 write index pointer, range 0 to 63

Table continues on the next page...

M_CAN_RXF0S field descriptions (continued)

Field	Description
16–17 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
18–23 F0GI	Rx FIFO 0 Get Index Rx FIFO 0 read index pointer, range 0 to 63.
24 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
25–31 F0FL	Rx FIFO 0 Fill Level Number of elements stored in Rx FIFO 0, range 0 to 64.

41.3.28 Rx FIFO 0 Acknowledge Register (M_CAN_RXF0A)

Address: 0h base + A8h offset = A8h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0																									F0AI						
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

M_CAN_RXF0A field descriptions

Field	Description
0–25 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
26–31 F0AI	Rx FIFO 0 Acknowledge Index After the CPU has read a message or a sequence of messages from Rx FIFO 0 it has to write the buffer index of the last element read from Rx FIFO 0 to F0AI. This will set the Rx FIFO 0 Get Index RXF0S[F0GI] to F0AI + 1 and update the FIFO 0 Fill Level RXF0S[F0FL].

41.3.29 Rx Buffer Configuration Register (M_CAN_RXBC)

Address: 0h base + ACh offset = ACh

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0															RBSA														0		
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	

M_CAN_RXBC field descriptions

Field	Description
0–15 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

Table continues on the next page...

M_CAN_RXBC field descriptions (continued)

Field	Description
16–29 RBSA	Rx Buffer Start Address Configures the start address of the Rx Buffers section in the Message RAM (32-bit word address). Also used to reference debug messages A, B, C. NOTE: This field has Protected Write status.
30–31 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

41.3.30 Rx FIFO 1 Configuration Register (M_CAN_RXF1C)

Address: 0h base + B0h offset = B0h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	F1OM	F1WM							0	F1S						
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	F1SA														0	
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

M_CAN_RXF1C field descriptions

Field	Description
0 F1OM	FIFO 1 Operation Mode FIFO 1 can be operated in blocking or in overwrite mode (see Rx FIFOs). NOTE: This field has Protected Write status. 0 FIFO 1 blocking mode 1 FIFO 1 overwrite mode
1–7 F1WM	Rx FIFO 1 Watermark NOTE: This field has Protected Write status. 0 Watermark interrupt disabled 1–64 Level for Rx FIFO 1 watermark interrupt (IR[RF1W]) >64 Watermark interrupt disabled
8 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

Table continues on the next page...

M_CAN_RXF1C field descriptions (continued)

Field	Description
9–15 F1S	Rx FIFO 1 Size The Rx FIFO 1 elements are indexed from 0 to F1S - 1. NOTE: This field has Protected Write status. 0 No Rx FIFO 1 1-64 Number of Rx FIFO 1 elements >64 Values greater than 64 are interpreted as 64
16–29 F1SA	Rx FIFO 1 Start Address Start address of Rx FIFO 1 in Message RAM (32-bit word address, see Message RAM). NOTE: This field has Protected Write status.
30–31 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

41.3.31 Rx FIFO 1 Status Register (M_CAN_RXF1S)

Address: 0h base + B4h offset = B4h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	DMS			0			RF1L	F1F		0					F1PI	
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	Reserved							F1GI		0					F1FL	
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

M_CAN_RXF1S field descriptions

Field	Description
0–1 DMS	Debug Message Status 00 Idle state, wait for reception of debug messages, DMA request is cleared 01 Debug message A received 10 Debug messages A, B received 11 Debug messages A, B, C received, DMA request is set
2–5 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
6 RF1L	Rx FIFO 1 Message Lost This bit is a copy of interrupt flag IR[RF1L]. When IR[RF1L] is reset, this bit is also reset. NOTE: Overwriting the oldest message when RXF1C[F1OM] = 1 will not set this flag.

Table continues on the next page...

M_CAN_RXF1S field descriptions (continued)

Field	Description
	0 No Rx FIFO 1 message lost 1 Rx FIFO 1 message lost, also set after write attempt to Rx FIFO 1 of size zero
7 F1F	Rx FIFO 1 Full 0 Rx FIFO 1 not full 1 Rx FIFO 1 full
8–9 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
10–15 F1PI	Rx FIFO 1 Put Index Rx FIFO 1 write index pointer, range 0 to 63.
16–17 Reserved	This field is reserved.
18–23 F1GI	Rx FIFO 1 Get Index Rx FIFO 1 read index pointer, range 0 to 63.
24 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
25–31 F1FL	Rx FIFO 1 Fill Level Number of elements stored in Rx FIFO 1, range 0 to 64.

41.3.32 Rx FIFO 1 Acknowledge Register (M_CAN_RXF1A)

Address: 0h base + B8h offset = B8h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0																									F1AI						
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

M_CAN_RXF1A field descriptions

Field	Description
0–25 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
26–31 F1AI	Rx FIFO 1 Acknowledge Index After the CPU has read a message or a sequence of messages from Rx FIFO 1 it has to write the buffer index of the last element read from Rx FIFO 1 to F1AI. This will set the Rx FIFO 1 Get Index RXF1S[F1GI] to F1AI + 1 and update the FIFO 1 Fill Level RXF1S[F1FL].

41.3.33 Rx Buffer / FIFO Element Size Configuration Register (M_CAN_RXESC)

Configures the number of data bytes belonging to an Rx Buffer / Rx FIFO element. Data field sizes >8 bytes are intended for CAN FD operation only.

NOTE

In case the data field size of an accepted CAN frame exceeds the data field size configured for the matching Rx Buffer or Rx FIFO, only the number of bytes as configured by RXESC are stored to the Rx Buffer resp. Rx FIFO element. The rest of the frame's data field is ignored.

Address: 0h base + BCh offset = BCh

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0				RBDS				0	F1DS				0	F0DS	
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

M_CAN_RXESC field descriptions

Field	Description
0–20 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
21–23 RBDS	Rx Buffer Data Field Size NOTE: This field has Protected Write status. 000 8 byte data field 001 12 byte data field 010 16 byte data field 011 20 byte data field 100 24 byte data field 101 32 byte data field 110 48 byte data field 111 64 byte data field
24 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
25–27 F1DS	Rx FIFO 1 Data Field Size NOTE: This field has Protected Write status.

Table continues on the next page...

M_CAN_RXESC field descriptions (continued)

Field	Description
	000 8 byte data field 001 12 byte data field 010 16 byte data field 011 20 byte data field 100 24 byte data field 101 32 byte data field 110 48 byte data field 111 64 byte data field
28 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
29–31 F0DS	Rx FIFO 0 Data Field Size NOTE: This field has Protected Write status. 000 8 byte data field 001 12 byte data field 010 16 byte data field 011 20 byte data field 100 24 byte data field 101 32 byte data field 110 48 byte data field 111 64 byte data field

41.3.34 Tx Buffer Configuration Register (M_CAN_TXBC)

NOTE

Be aware that the sum of TFQS and NDTB may be not greater than 32. There is no check for erroneous configurations. The Tx Buffers section in the Message RAM starts with the dedicated Tx Buffers.

Address: 0h base + C0h offset = C0h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	TFQM	TFQS						0	NDTB						
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	TBSA														0	
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

M_CAN_TXBC field descriptions

Field	Description
0 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
1 TFQM	Tx FIFO/Queue Mode NOTE: This field has Protected Write status. 0 Tx FIFO operation 1 Tx Queue operation
2–7 TFQS	Transmit FIFO/Queue Size NOTE: This field has Protected Write status. 0 No Tx FIFO/Queue 1–32 Number of Tx Buffers used for Tx FIFO/Queue >32 Values greater than 32 are interpreted as 32
8–9 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

Table continues on the next page...

M_CAN_TXBC field descriptions (continued)

Field	Description
10–15 NDTB	Number of Dedicated Transmit Buffers NOTE: This field has Protected Write status. 0 No Dedicated Tx Buffers 1–32 Number of Dedicated Tx Buffers >32 Values greater than 32 are interpreted as 32
16–29 TBSA	Tx Buffers Start Address Start address of Tx Buffers section in Message RAM (32-bit word address, see Message RAM). NOTE: This field has Protected Write status.
30–31 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

41.3.35 Tx FIFO/Queue Status Register (M_CAN_TXFQS)

The Tx FIFO/Queue status is related to the pending Tx requests listed in register TXBRP. Therefore the effect of Add/Cancellation requests may be delayed due to a running Tx scan (TXBRP not yet updated).

NOTE

In case of mixed configurations where dedicated Tx Buffers are combined with a Tx FIFO or a Tx Queue, the Put and Get Indices indicate the number of the Tx Buffer starting with the first dedicated Tx Buffers.

Example: For a configuration of 12 dedicated Tx Buffers and a Tx FIFO of 20 Buffers a Put Index of 15 points to the fourth buffer of the Tx FIFO.

Address: 0h base + C4h offset = C4h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0									TFQF		TFQPI				
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	
R	0			TFGI						0		TFFL					
W																	
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	

M_CAN_TXFQS field descriptions

Field	Description
0–9 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
10 TFQF	Tx FIFO/Queue Full 0 Tx FIFO/Queue not full 1 Tx FIFO/Queue full
11–15 TFQPI	Tx FIFO/Queue Put Index Tx FIFO/Queue write index pointer, range 0 to 31.
16–18 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
19–23 TFGI	Tx FIFO Get Index Tx FIFO read index pointer, range 0 to 31. Read as zero when Tx Queue operation is configured (TXBC[TFQM] = 1).
24–25 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
26–31 TFFL	Tx FIFO Free Level Number of consecutive free Tx FIFO elements starting from TFGI, range 0 to 32. Read as zero when Tx Queue operation is configured (TXBC[TFQM] = 1).

41.3.36 Tx Buffer Element Size Configuration (M_CAN_TXESC)

Configures the number of data bytes belonging to a Tx Buffer element. Data field sizes > 8 bytes are intended for CAN FD operation only.

Address: 0h base + C8h offset = C8h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0																												TBDS			
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	

M_CAN_TXESC field descriptions

Field	Description
0–28 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
29–31 TBDS	Tx Buffer Data Field Size NOTE: In case the data length code DLC of a Tx Buffer element is configured to a value higher than the Tx Buffer data field size TXESC[TBDS], the bytes not defined by the Tx Buffer are transmitted as 0xCC (padding bytes). NOTE: This field has Protected Write status.

Table continues on the next page...

M_CAN_TXESC field descriptions (continued)

Field	Description
000	8 byte data field
001	12 byte data field
010	16 byte data field
011	20 byte data field
100	24 byte data field
101	32 byte data field
110	48 byte data field
111	64 byte data field

41.3.37 Tx Buffer Request Pending Register (M_CAN_TXBRP)**NOTE**

TXBRP bits which are set while a Tx scan is in progress are not considered during this particular Tx scan. In case a cancellation is requested for such a Tx Buffer, this Add Request is cancelled immediately, the corresponding TXBRP bit is reset.

Address: 0h base + CCh offset = CCh

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	TRP																															
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

M_CAN_TXBRP field descriptions

Field	Description
0–31 TRP	Transmission Request Pending Each Tx Buffer has its own Transmission Request Pending bit. The bits are set via register TXBAR. The bits are reset after a requested transmission has completed or has been cancelled via register TXBCR. TXBRP bits are set only for those Tx Buffers configured via TXBC. After a TXBRP bit has been set, a Tx scan (see Tx Handling) is started to check for the pending Tx request with the highest priority (Tx Buffer with lowest Message ID). A cancellation request resets the corresponding transmission request pending bit of register TXBRP. In case a transmission has already been started when a cancellation is requested, this is done at the end of the transmission, regardless whether the transmission was successful or not. The cancellation request bits are reset directly after the corresponding TXBRP bit has been reset. After a cancellation has been requested, a finished cancellation is signaled via TXBCF • after successful transmission together with the corresponding TXBTO bit • when the transmission has not yet been started at the point of cancellation • when the transmission has been aborted due to lost arbitration • when an error occurred during frame transmission

M_CAN_TXBRP field descriptions (continued)

Field	Description
	In DAR mode all transmissions are automatically cancelled if they are not successful. The corresponding TXBCF bit is set for all unsuccessful transmissions.
0	No transmission request pending
1	Transmission request pending

41.3.38 Tx Buffer Add Request Register (M_CAN_TXBAR)**NOTE**

If an add request is applied for a Tx Buffer with pending transmission request (corresponding TXBRP bit already set), this add request is ignored.

Address: 0h base + D0h offset = D0h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R																																
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

M_CAN_TXBAR field descriptions

Field	Description
0–31 AR	Add Request Each Tx Buffer has its own Add Request bit. Writing 1 will set the corresponding Add Request bit; writing 0 has no impact. This enables the CPU to set transmission requests for multiple Tx Buffers with one write to TXBAR. TXBAR bits are set only for those Tx Buffers configured via TXBC. When no Tx scan is running, the bits are reset immediately, else the bits remain set until the Tx scan process has completed. 0 No transmission request added 1 Transmission requested added

41.3.39 Tx Buffer Cancellation Request Register (M_CAN_TXBCR)

Address: 0h base + D4h offset = D4h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R																																
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

M_CAN_TXBCR field descriptions

Field	Description
0–31 CR	Cancellation Request Each Tx Buffer has its own Cancellation Request bit. Writing 1 will set the corresponding Cancellation Request bit; writing 0 has no impact. This enables the CPU to set cancellation requests for multiple Tx Buffers with one write to TXBCR. TXBCR bits are set only for those Tx Buffers configured via TXBC. The bits remain set until the corresponding bit of TXBRP is reset. 0 No cancellation pending 1 Cancellation pending

41.3.40 Tx Buffer Transmission Occurred Register (M_CAN_TXBTO)

Address: 0h base + D8h offset = D8h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R																TO																
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

M_CAN_TXBTO field descriptions

Field	Description
0–31 TO	Transmission Occurred Each Tx Buffer has its own Transmission Occurred bit. The bits are set when the corresponding TXBRP bit is cleared after a successful transmission. The bits are reset when a new transmission is requested by writing 1 to the corresponding bit of register TXBAR. 0 No transmission occurred 1 Transmission occurred

41.3.41 Tx Buffer Cancellation Finished Register (M_CAN_TXBCF)

Address: 0h base + DCh offset = DCh

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	CF																															
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	

M_CAN_TXBCF field descriptions

Field	Description
0–31 CF	Cancellation Finished

M_CAN_TXBCF field descriptions (continued)

Field	Description
	Each Tx Buffer has its own Cancellation Finished bit. The bits are set when the corresponding TXBRP bit is cleared after a cancellation was requested via TXBCR. In case the corresponding TXBRP bit was not set at the point of cancellation, CF is set immediately. The bits are reset when a new transmission is requested by writing 1 to the corresponding bit of register TXBAR.
0	No transmit buffer cancellation
1	Transmit buffer cancellation finished

41.3.42 Tx Buffer Transmission Interrupt Enable Register (M_CAN_TXBTIE)

Address: 0h base + E0h offset = E0h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R																																
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

M_CAN_TXBTIE field descriptions

Field	Description
0–31 TIE	Transmission Interrupt Enable Each Tx Buffer has its own Transmission Interrupt Enable bit. 0 Transmission interrupt disabled 1 Transmission interrupt enable

41.3.43 Tx Buffer Cancellation Finished Interrupt Enable Register (M_CAN_TXBCIE)

Address: 0h base + E4h offset = E4h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R																																
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

M_CAN_TXBCIE field descriptions

Field	Description
0–31 CFIE	Cancellation Finished Interrupt Enable Each Tx Buffer has its own Cancellation Finished Interrupt Enable bit. 0 Cancellation finished interrupt disabled 1 Cancellation finished interrupt enabled

41.3.44 Tx Event FIFO Configuration Register (M_CAN_TXEFC)

Address: 0h base + F0h offset = F0h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	EFWM							0	EFS							EFSA															0
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

M_CAN_TXEFC field descriptions

Field	Description
0–1 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
2–7 EFWM	Event FIFO Watermark NOTE: This field has Protected Write status. 0 Watermark interrupt disabled 1-32 Level for Tx Event FIFO watermark interrupt (IR[TEFW]) >32 Watermark interrupt disabled
8–9 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
10–15 EFS	Event FIFO Size The Tx Event FIFO elements are indexed from 0 to EFS - 1 NOTE: This field has Protected Write status. 0 Tx Event FIFO disabled 1-32 Number of Tx Event FIFO elements >32 Values greater than 32 are interpreted as 32
16–29 EFSA	Event FIFO Start Address Start address of Tx Event FIFO in Message RAM (32-bit word address, Message RAM). NOTE: This field has Protected Write status.
30–31 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

41.3.45 Tx Event FIFO Status Register (M_CAN_TXEFS)

Address: 0h base + F4h offset = F4h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	
R	0						TEFL	EFF	0			EFPI					
W																	
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	

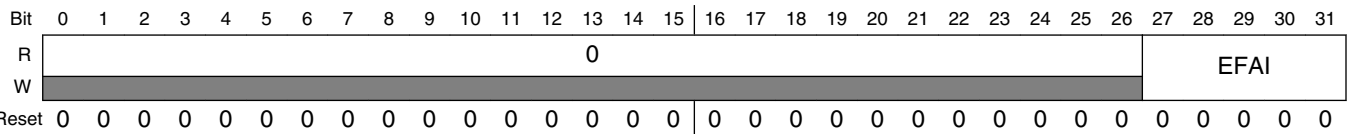
Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0			EFGI					0		EFFL					
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

M_CAN_TXEFS field descriptions

Field	Description
0–5 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
6 TEFL	Tx Event FIFO Element Lost This bit is a copy of interrupt flag IR[TEFL]. When IR[TEFL] is reset, this bit is also reset. 0 No Tx Event FIFO element lost 1 Tx Event FIFO element lost, also set after write attempt to Tx Event FIFO of size zero
7 EFF	Event FIFO Full 0 Tx Event FIFO not full 1 Tx Event FIFO full
8–10 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
11–15 EFPI	Event FIFO Put Index Tx Event FIFO write index pointer, range 0 to 31.
16–18 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
19–23 EFGI	Event FIFO Get Index Tx Event FIFO read index pointer, range 0 to 31.
24–25 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
26–31 EFFL	Event FIFO Fill Level Number of elements stored in Tx Event FIFO, range 0 to 32.

41.3.46 Tx Event FIFO Acknowledge Register (M_CAN_TXEFA)

Address: 0h base + F8h offset = F8h



M_CAN_TXEFA field descriptions

Field	Description
0–26 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
27–31 EFAI	Event FIFO Acknowledge Index After the CPU has read an element or a sequence of elements from the Tx Event FIFO it has to write the index of the last element read from Tx Event FIFO to EFAI. This will set the Tx Event FIFO Get Index TXEFS[EFGI] to EFAI + 1 and update the Event FIFO Fill Level TXEFS[EFFL].

41.4 Message RAM

For storage of Rx/Tx messages and for storage of the filter configuration a single- or dual-ported Message RAM has to be connected to the M_CAN module.

NOTE

In case the MessageRAMis equipped with parity or ECC functionality, it is recommended to initialize the Message RAM after hardware reset by writing e.g. 0x00000000 to each Message RAM word to create valid parity/ECC checksums. This avoids that reading from uninitialized Message RAM sections will activate interrupt IR[BEC] (Bit Error Corrected) or IR[BEU] (Bit Error Uncorrected).

The Message RAM has a width of 32 bits. In case parity checking or ECC is used a respective number of bits has to be added to each word.

When operated in CAN FD mode the required Message RAM size strongly depends on the element size configured for Rx FIFO0, Rx FIFO1, Rx Buffers, and Tx Buffers via RXESC[F0DS], RXESC[F1DS], RXESC[RBDS], and TXESC[TBDS].

The M_CAN module can be configured to allocate up to 4352 words in the Message RAM. For the actual amount of Message RAM on this chip, see the chip-specific M_CAN information.

It is not necessary to configure each of the sections listed in the following figure, and there is no restriction with respect to the sequence of the sections.

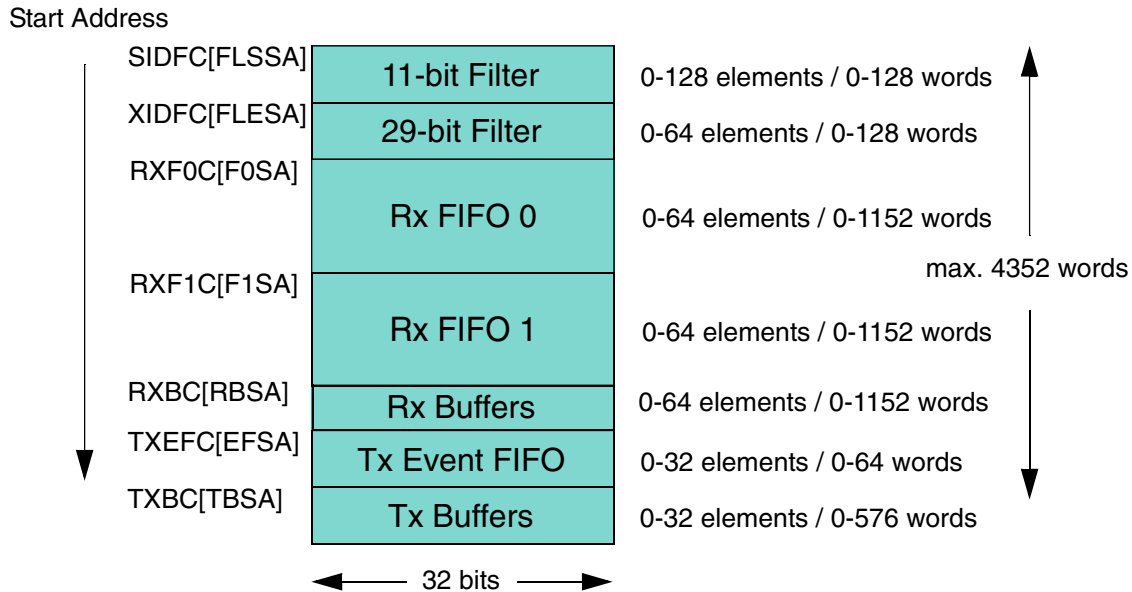


Figure 41-49. Message RAM configuration

When the M_CAN addresses the Message RAM it addresses 32-bit words, not single bytes. The configurable start addresses are 32-bit word addresses i.e. only bits 15 to 2 are evaluated, the two least significant bits are ignored.

NOTE

The M_CAN does not check for erroneous configuration of the Message RAM. Especially the configuration of the start addresses of the different sections and the number of elements of each section has to be done carefully to avoid falsification or loss of data.

41.4.1 Rx Buffer and FIFO Element

Up to 64 Rx Buffers and two Rx FIFOs can be configured in the Message RAM. Each Rx FIFO section can be configured to store up to 64 received messages. The structure of a Rx Buffer / FIFO element is shown in the following table. The element size can be configured for storage of CAN FD messages with up to 64 bytes data field via register RXESC.

Table 41-49. Rx Buffer and FIFO Element

	31			24		23			16			15			8		7			0	
R0	E	X	R	ID[28:0]																	
	S	T	T																		
	I	D	R																		
R1	ANMF	FIDX[6:0]				res		F	D	F	B	R	DLC[3:0]		RXTS[15:0]						
R2	DB3[7:0]				DB2[7:0]				DB1[7:0]				DB0[7:0]								
R3	DB7[7:0]				DB6[7:0]				DB5[7:0]				DB4[7:0]								
	...				...				...				...								
Rn	DBm[7:0]				DBm-1[7:0]				DBm-2[7:0]				DBm-3[7:0]								

Table 41-50. Rx Buffer and FIFO Element Descriptions

R0 Bit 31	ESI: Error State Indicator 0 Transmitting node is error active 1 Transmitting node is error passive
R0 Bit 30	XTD: Extended Identifier Signals to the Host whether the received frame has a standard or extended identifier. 0 11-bit standard identifier 1 29-bit standard identifier
R0 Bit 29	RTR: Remote Transmission Request Signals to the Host whether the received frame is a data frame or a remote frame. 0 Received frame is a data frame 1 Received frame is a remote frame NOTE: There are no remote frames in CAN FD format. In CAN FD frames (FDF = 1'), the dominant RRS (Remote Request Substitution) bit replaces bit RTR (Remote Transmission Request).
R0 Bits 28:0	ID[28:0]: Identifier Standard or extended identifier depending on bit XTD. A standard identifier is stored into ID[28:18].
R1 Bit 31	ANMF: Accepted Non-matching Frame Acceptance of non-matching frames may be enabled via GFC[ANFS] and GFC[ANFE]. 0 Received frame matching filter index FIDX 1 Received frame did not match any Rx filter element
R1 Bits 30:24	FIDX[6:0]: Filter Index 0-127 Index of matching Rx acceptance filter element (invalid if ANMF = 1). Range is 0 to SIDFC[LSS] - 1 resp. XIDFC[LSE] - 1.
R1 Bit 21	FDF: FD Format 0 Standard frame format 1 CAN FD frame format (new DLC-coding and CRC)

Table continues on the next page...

Table 41-50. Rx Buffer and FIFO Element Descriptions (continued)

R1 Bit 20	BRS: Bit Rate Switch 0 Frame received without bit rate switching 1 Frame received with bit rate switching
R1 Bits 19:16	DLC[3:0]: Data Length Code 0-8 CAN + CAN FD: received frame has 0-8 data bytes 9-15 CAN: received frame has 8 data bytes 9-15 CAN FD: received frame has 12/16/20/24/32/48/64 data bytes
R1 Bits 15:0	RXTS[15:0]: Rx Timestamp Timestamp Counter value captured on start of frame reception. Resolution depending on configuration of the Timestamp Counter Prescaler TSCC[TCP].
R2 Bits 31:24	DB3[7:0]: Data Byte 3
R2 Bits 23:16	DB2[7:0]: Data Byte 2
R2 Bits 15:8	DB1[7:0]: Data Byte 1
R2 Bits 7:0	DB0[7:0]: Data Byte 0
R3 Bits 31:24	DB7[7:0]: Data Byte 7
R3 Bits 23:16	DB6[7:0]: Data Byte 6
R3 Bits 15:8	DB5[7:0]: Data Byte 5
R3 Bits 7:0	DB4[7:0]: Data Byte 4
...	...
Rn Bits 31:24	DBm[7:0]: Data Byte m
Rn Bits 23:16	DBm-1[7:0]: Data Byte m-1
Rn Bits 15:8	DBm-2[7:0]: Data Byte m-2
Rn Bits 7:0	DBm-3[7:0]: Data Byte m-3

NOTE

Depending on the configuration of the element size (RXESC), between two and sixteen 32-bit words ($R_n = 3 \dots 17$) are used for storage of a CAN message's data field.

41.4.2 Tx Buffer Element

The Tx Buffers section can be configured to hold dedicated Tx Buffers as well as a Tx FIFO / Tx Queue. In case that the Tx Buffers section is shared by dedicated Tx buffers and a Tx FIFO / Tx Queue, the dedicated Tx Buffers start at the beginning of the Tx Buffers section followed by the buffers assigned to the Tx FIFO or Tx Queue. The Tx Handler distinguishes between dedicated Tx Buffers and Tx FIFO / Tx Queue by evaluating the Tx Buffer configuration TXBC.TFQS and TXBC.NDTB. The element size can be configured for storage of CAN FD messages with up to 64 bytes data field via register TXESC.

Table 41-51. Tx Buffer Element

	31	24	23	16	15	8	7	0
T0	E SI	X T D	R T R	ID[28:0]				
T1	MM[7:0]			E F C	re s	F D F	B R S	DLC[3:0] res
T2	DB3[7:0]			DB2[7:0]			DB1[7:0]	
T3	DB7[7:0]			DB6[7:0]			DB5[7:0]	
...	...			...			...	
Tn	DBm[7:0]			DBm-1[7:0]			DBm-2[7:0]	
							DBm-3[7:0]	

Table 41-52. Tx Buffer Element Description

T0 Bit 31	ESI: Error State Indicator 0 ESI bit in CAN FD format depends only on error passive flag 1 ESI bit in CAN FD format transmitted recessive NOTE: The ESI bit of the transmit buffer is OR-ed with the error passive flag to decide the value of the ESI bit in the transmitted FD frame. As required by the CAN FD protocol specification, an error active node may optionally transmit the ESI bit recessive, but an error passive node will always transmit the ESI bit recessive.
T0 Bit 30	XTD: Extended Identifier 0 11-bit standard identifier 1 29-bit extended identifier
T0 Bit 29	RTR: Remote Transmission Request 0 Transmit data frame 1 Transmit remote frame NOTE: When RTR = 1, the M_CAN transmits a remote frame according to ISO11898-1, even if CCCR[FDOE] enables the transmission in CAN FD format.
T0 Bit 28:0	ID[28:0]: Identifier Standard or extended identifier depending on bit XTD. A standard identifier has to be written to ID[28:18].
T1 Bits 31:24	MM[7:0]: Message Marker Written by CPU during Tx Buffer configuration. Copied into Tx Event FIFO element for identification of Tx message status.
T1 Bit 23	EFC: Event FIFO Control 0 Do not store Tx events 1 Store Tx events
T1 Bit 21	FDF: FD Format 0 Frame transmitted in Classic CAN format 1 Frame transmitted in CAN FD format

Table continues on the next page...

Table 41-52. Tx Buffer Element Description (continued)

T1 Bit 20	BRS: Bit Rate Switching 0 CAN FD frames transmitted without bit rate switching 1 CAN FD frames transmitted with bit rate switching NOTE: Bits ESL, FDF, and BRS are only evaluated when CAN FD operation is enabled CCCR[FDOE] = 1. Bit BRS is only evaluated when in addition CCCR[BRSE] = 1.
T1 Bits 19:16	DLC[3:0]: Data Length Code 0-8 CAN + CAN FD: Transmit frame has 0-8 data bytes 9-15 CAN: Transmit frame has 8 data bytes 9-15 CAN FD: Transmit frame has 12/16/20/24/32/48/64 data bytes
T2 Bits 31:24	DB3[7:0]: Data Byte 3
T2 Bits 23:16	DB2[7:0]: Data Byte 2
T2 Bits 15:8	DB1[7:0]: Data Byte 1
T2 Bits 7:0	DB0[7:0]: Data Byte 0
T3 Bits 31:24	DB7[7:0]: Data Byte 7
T3 Bits 23:16	DB6[7:0]: Data Byte 6
T3 Bits 15:8	DB5[7:0]: Data Byte 5
T3 Bits 7:0	DB4[7:0]: Data Byte 4
...	...
Tn Bits 31:24	DBm[7:0]: Data Byte m
Tn Bits 23:16	DBm-1[7:0]: Data Byte m-1
Tn Bits 15:8	DBm-2[7:0]: Data Byte m-2
Tn Bits 7:0	DBm-3[7:0]: Data Byte m-3

NOTE

Depending on the configuration of the element size (TXESC), between two and sixteen 32-bit words (Tn = 3 ..17) are used for storage of a CAN message's data field.

41.4.3 Tx Event FIFO Element

Each element stores information about transmitted messages. By reading the Tx Event FIFO the Host CPU gets this information in the order the messages were transmitted. Status information about the Tx Event FIFO can be obtained from register TXEFS.

Table 41-53. Tx Event FIFO Element

	31	24	23	16	15	8	7	0
E0	E SI	X T D	R T R	ID[28:0]				

Table continues on the next page...

Table 41-53. Tx Event FIFO Element (continued)

E1	MM[7:0]	ET[1:0]	F D F	B R S	DLC[3:0]	TXTS[15:0]

Table 41-54. Tx Event FIFO Element Description

E0 Bit 31	ESI: Error State Indicator 0 Transmitting node is error active 1 Transmitting node is error passive
E0 Bit 30	XTD: Extended Identifier 0 11-bit standard identifier 1 29-bit extended identifier
E0 Bit 29	RTR: Remote Transmission Request 0 Data frame transmitted 1 Remote frame transmitted
E0 Bits 28:0	ID[28:0]: Identifier Standard or extended identifier depending on bit XTD. A standard identifier is stored into ID[28:18].
E1 Bits 31:24	MM[7:0]: Message Marker Copied from Tx Buffer into Tx Event FIFO element for identification of Tx message status.
E1 Bit 23:22	ET[1:0]: Event Type 00 Reserved 01 Tx event 10 Transmission in spite of cancellation (always set for transmissions in DAR mode) 11 Reserved
E1 Bit 21	FDF: FD Format 0 Standard frame format 1 CAN FD frame format (new DLC-coding and CRC)
E1 Bit 20	BRS: Bit Rate Switch 0 Frame transmitted without bit rate switching 1 Frame transmitted with bit rate switching
E1 Bits 19:16	DLC[3:0]: Data Length Code 0-8 CAN + CAN FD: Frame with 0-8 data bytes transmitted 9-15 CAN: Frame with 8 data bytes transmitted 9-15 CAN FD: frame with 12/16/20/24/32/48/64 data bytes transmitted
E1 Bits 15:0	TXTS[15:0]: Tx Timestamp Timestamp Counter value captured on start of frame transmission. Resolution depending on configuration of the Timestamp Counter Prescaler TSCC.TCP.

41.4.4 Standard Message ID Filter Element

Up to 128 filter elements can be configured for 11-bit standard IDs. When accessing a Standard Message ID Filter element, its address is the Filter List Standard Start Address SIDFC[FLSSA] plus the index of the filter element (0...127).

Table 41-55. Standard Message ID Filter Element

	31	24	23	16	15	8	7	0
S0	SFT[1:0]	SFEC[2:0]	SFID1[10:0]			res	SFID2[10:0]	

Table 41-56. Standard Message ID Filter Element Field Description

S0 Bits 31:30	SFT[1:0]: Standard Filter Type 00 Range filter from SFID1 to SFID2 (SFID2 >= SFID1) 01 Dual ID filter for SFID1 or SFID2 10 Classic filter: SFID1 = filter, SFID2 = mask 11 Filter element disabled NOTE: With SFT = 11 the filter element is disabled and the acceptance filtering continues (same behavior as with SFEC = 000).
S0 Bits 29:27	SFEC[2:0]: Standard Filter Element Configuration All enabled filter elements are used for acceptance filtering of standard frames. Acceptance filtering stops at the first matching enabled filter element or when the end of the filter list is reached. If SFEC = 100, 101, or 110 a match sets interrupt flag IR[HPM] and, if enabled, an interrupt is generated. In this case register HPMS is updated with the status of the priority match. 000 Disable filter element 001 Store in Rx FIFO 0 if filter matches 010 Store in Rx FIFO 1 if filter matches 011 Reject ID if filter matches 100 Set priority if filter matches 101 Set priority and store in FIFO 0 if filter matches 110 Set priority and store in FIFO 1 if filter matches 111 Store into Rx Buffer or as debug message, configuration of SFT[1:0] ignored
S0 Bits 26:16	SFID1[10:0]: Standard Filter ID 1 First ID of standard ID filter element. When filtering for Rx Buffers or for debug messages this field defines the ID of a standard message to be stored. The received identifiers must match exactly, no masking mechanism is used.

Table continues on the next page...

Table 41-56. Standard Message ID Filter Element Field Description (continued)

S0 Bits 10:0	SFID2[10:0]: Standard Filter ID 2 Overall, this bit field has a different meaning depending on the configuration of SFEC: SFEC = 001...110 Second ID of standard ID filter element SFEC = 111 Filter for Rx Buffers or for debug messages
	SFID2[10:9]: Decides whether the received message is stored into an Rx Buffer or treated as message A, B, or C of the debug message sequence. 00 Store message into an Rx Buffer 01 Debug Message A 10 Debug Message B 11 Debug Message C
	SFID2[8:6]: Is used to control the M_CAN filter event pins at the Extension Interface. A one at the respective bit position enables generation of a pulse at the related filter event pin with the duration of one Host clock period in case the filter matches.
	SFID2[5:0]: Defines the offset to the Rx Buffer Start Address RXBC.RBSA for storage of a matching message.

41.4.5 Extended Message ID Filter Element

Up to 64 filter elements can be configured for 29-bit extended IDs. When accessing an Extended Message ID Filter element, its address is the Filter List Extended Start Address XIDFC[FLESA] plus two times the index of the filter element (0...63).

Table 41-57. Extended Message ID Filter Element

	31	24	23	16	15	8	7	0
F0	EFEC[2:0]	EFID1[28:0]						
F1	EFT[1:0]	res	EFID2[28:0]					

Table 41-58. Extended Message ID Filter Element Field Description

F0 Bits 31:29	EFEC[2:0]: Extended Filter Element Configuration All enabled filter elements are used for acceptance filtering of extended frames. Acceptance filtering stops at the first matching enabled filter element or when the end of the filter list is reached. If EFEC = 100, 101, or 110 a match sets interrupt flag IR[HPM] and, if enabled, an interrupt is generated. In this case register HPMS is updated with the status of the priority match. 000 Disable filter element 001 Store in Rx FIFO 0 if filter matches 010 Store in Rx FIFO 1 if filter matches 011 Reject ID if filter matches 100 Set priority if filter matches 101 Set priority and store in FIFO 0 if filter matches 110 Set priority and store in FIFO 1 if filter matches 111 Store into Rx Buffer or as debug message, configuration of EFT[1:0] ignored
F0 Bits 28:0	EFID1[28:0]: Extended Filter ID 1 First ID of extended ID filter element. When filtering for Rx Buffers or for debug messages this field defines the ID of an extended message to be stored. The received identifiers must match exactly, only XIDAM masking mechanism (see Extended Message ID Filtering) is used.
F1 Bits 31:30	EFT[1:0]: Extended Filter Type 00 Range filter from EFID1 to EFID2 (EFID2 >= EFID1) 01 Dual ID filter for EFID1 or EFID2 10 Classic filter: EFID1 = filter, EFID2 = mask 11 Range filter from EFID1 to EFID2 (EFID2 >= EFID1), XIDAM mask not applied
F1 Bits 28:0	EFID2[28:0]: Extended Filter ID 2 Overall, this bit field has a different meaning depending on the configuration of EFEC: EFEC = 001...110 Second ID of extended ID filter element EFEC = 111 Filter for Rx Buffers or for debug messages EFID2[10:9]: Decides whether the received message is stored into an Rx Buffer or treated as message A, B, or C of the debug message sequence. 00 Store message into an Rx Buffer 01 Debug Message A 10 Debug Message B 11 Debug Message C EFID2[8:6]: Is used to control the filter event pins at the Extension Interface. A one at the respective bit position enables generation of a pulse at the related filter event pin with the duration of one Host clock period in case the filter matches. EFID2[5:0]: Defines the offset to the Rx Buffer Start Address RXBC[RBSA] for storage of a matching message.

41.5 Functional Description

41.5.1 Operating Modes

41.5.1.1 Software Initialization

Software initialization is started by setting bit CCCR[INIT], either by software or by a hardware reset, when an uncorrected bit error was detected in the Message RAM, or by going Bus_Off. While CCCR[INIT] is set, message transfer from and to the CAN bus is stopped, the status of the CAN bus output M_CAN_Tx is recessive (HIGH). The counters of the Error Management Logic EML are unchanged. Setting CCCR[INIT] does not change any configuration register. Resetting CCCR[INIT] finishes the software initialization. Afterwards the Bit Stream Processor BSP synchronizes itself to the data transfer on the CAN bus by waiting for the occurrence of a sequence of 11 consecutive recessive bits (= Bus_Idle) before it can take part in bus activities and start the message transfer.

Access to the M_CAN configuration registers is only enabled when both bits CCCR[INIT] and CCCR[CCE] are set (protected write).

CCCR[CCE] can only be set/reset while CCCR[INIT] = 1. CCCR[CCE] is automatically reset when CCCR[INIT] is reset.

The following registers are reset when CCCR[CCE] is set

- HPMS – High Priority Message Status
- RXF0S – Rx FIFO 0 Status
- RXF1S – Rx FIFO 1 Status
- TXFQS – Tx FIFO/Queue Status
- TXBRP – Tx Buffer Request Pending
- TXBTO – Tx Buffer Transmission Occurred
- TXBCF – Tx Buffer Cancellation Finished
- TXEFS – Tx Event FIFO Status

The Timeout Counter value TOCV[TOC] is preset to the value configured by TOCC[TOP] when CCCR[CCE] is set.

In addition the state machines of the Tx Handler and Rx Handler are held in idle state while $\text{CCCR}[\text{CCE}] = 1$.

The following registers are only writable while $\text{CCCR}[\text{CCE}] = 0$

- TXBAR – Tx Buffer Add Request
- TXBCR – Tx Buffer Cancellation Request

$\text{CCCR}[\text{TEST}]$ and $\text{CCCR}[\text{MON}]$ can only be set by the Host while $\text{CCCR}[\text{INIT}] = 1$ and $\text{CCCR}[\text{CCE}] = 1$. Both bits may be reset at any time. $\text{CCCR}[\text{DAR}]$ can only be set/reset while $\text{CCCR}[\text{INIT}] = 1$ and $\text{CCCR}[\text{CCE}] = 1$.

NOTE

In case the Message RAM is equipped with parity or ECC functionality, it is recommended to initialize the Message RAM after hardware reset by writing e.g. 0x00000000 to each Message RAM word to create valid parity/ECC checksums. This avoids that reading from uninitialized Message RAM sections will activate interrupt IR.BEC (Bit Error Corrected) or IR.BEU (Bit Error Uncorrected).

41.5.1.2 Normal Operation

Once the M_CAN is initialized and $\text{CCCR}.\text{INIT}$ is reset to zero, the M_CAN synchronizes itself to the CAN bus and is ready for communication.

After passing the acceptance filtering, received messages including Message ID and DLC are stored into a dedicated Rx Buffer or into Rx FIFO 0 or Rx FIFO 1.

For messages to be transmitted dedicated Tx Buffers and/or a Tx FIFO or a Tx Queue can be initialized or updated. Automated transmission on reception of remote frames is not implemented.

41.5.1.3 CAN FD Operation

There are two variants in the CAN FD frame transmission, first the CAN FD frame without bit rate switching. The second variant is the CAN FD frame where control field, data field, and CRC field are transmitted with a higher bit rate than the beginning and the end of the frame.

The previously reserved bit in CAN frames with 11-bit identifiers and the first previously reserved bit in CAN frames with 29-bit identifiers will now be decoded as FDF bit. FDF = recessive signifies a CAN FD frame, FDF = dominant signifies a Classic CAN frame. In a CAN FD frame, the two bits following FDF, res and BRS, decide whether the bit rate inside of this CAN FD frame is switched. A CAN FD bit rate switch is signified by res = dominant and BRS = recessive. The coding of res = recessive is reserved for future expansion of the protocol. In case the M_CAN receives a frame with FDF = recessive and res = recessive, it will signal a Protocol Exception Event by setting bit PSR[PXE]. When Protocol Exception Handling is enabled (CCCR[PXHD] = 0), this causes the operation state to change from Receiver (PSR[ACT] = 10) to Integrating (PSR[ACT] = 00) at the next sample point. In case Protocol Exception Handling is disabled (CCCR[PXHD] = 1), the M_CAN will treat a recessive res bit as an form error and will respond with an error frame.

CAN FD operation is enabled by programming CCCR[FDOE]. In case CCCR[FDOE] = 1, transmission and reception of CAN FD frames is enabled. Transmission and reception of Classic CAN frames is always possible. Whether a CAN FD frame or a Classic CAN frame is transmitted can be configured via bit FDF in the respective Tx Buffer element. With CCCR[FDOE] = 0, received frames are interpreted as Classic CAN frames, which leads to the transmission of an error frame when receiving a CAN FD frame. When CAN FD operation is disabled, no CAN FD frames are transmitted even if bit FDF of a Tx Buffer element is set. CCCR[FDOE] and CCCR[BRSE] can only be changed while CCCR[INIT] and CCCR[CCE] are both set.

With CCCR[FDOE] = 0, the setting of bits FDF and BRS is ignored and frames are transmitted in Classic CAN format. With CCCR[FDOE] = 1 and CCCR[BRSE] = 0, only bit FDF of a Tx Buffer element is evaluated. With CCCR[FDOE] = 1 and CCCR[BRSE] = 1, transmission of CAN FD frames with bit rate switching is enabled. All Tx Buffer elements with bits FDF and BRS set are transmitted in CAN FD format with bit rate switching.

A mode change during CAN operation is only recommended under the following conditions:

- The failure rate in the CAN FD data phase is significant higher than in the CAN FD arbitration phase. In this case disable the CAN FD bit rate switching option for transmissions.
- During system startup all nodes are transmitting Classic CAN messages until it is verified that they are able to communicate in CAN FD format. If this is true, all nodes switch to CAN FD operation.

- Wake-up messages in CAN Partial Networking have to be transmitted in Classic CAN format.
- End-of-line programming in case not all nodes are CAN FD capable. Non CAN FD nodes are held in Silent mode until programming has completed. Then all nodes switch back to Classic CAN communication.

In the CAN FD format, the coding of the DLC differs from the standard CAN format. The DLC codes 0 to 8 have the same coding as in standard CAN, the codes 9 to 15, which in standard CAN all code a data field of 8 bytes, are coded according to the following table.

Table 41-59. Coding of DLC in CAN FD

DLC	9	10	11	12	13	14	15
Number of data bytes	12	16	20	24	32	48	64

In CAN FD frames, the bit timing will be switched inside the frame, after the BRS (Bit Rate Switch) bit, if this bit is recessive. Before the BRS bit, in the CAN FD arbitration phase, the standard CAN bit timing is used as defined by the Bit Timing and Prescaler Register BTP. In the following CAN FD data phase, the fast CAN bit timing is used as defined by the Fast Bit Timing and Prescaler Register FBTP. The bit timing is switched back from the fast timing at the CRC delimiter or when an error is detected, whichever occurs first.

The maximum configurable bit rate in the CAN FD data phase depends on the CAN clock frequency. Example: with a CAN clock frequency of 20 MHz and the shortest configurable bit time of 4 t_q , the bit rate in the data phase is 5 Mbit/s.

In both data frame formats, CAN FD long and CAN FD fast, the value of the bit ESI (Error Status Indicator) is determined by the transmitter's error state at the start of the transmission. If the transmitter is error passive, ESI is transmitted recessive, else it is transmitted dominant.

41.5.1.4 Transmitter Delay Compensation

During the data phase of a CAN FD transmission only one node is transmitting, all others are receivers. The length of the bus line has no impact. When transmitting via pin M_CAN_Tx the M_CAN receives the transmitted data from its local CAN transceiver via M_CAN_Rx pin. The received data is delayed by the transmitter delay. In case this delay is greater than TSEG1 (time segment before sample point), a bit error is detected.

In order to enable a data phase bit time that is even shorter than the transmitter delay, the delay compensation is introduced. Without transmitter delay compensation, the bit rate in the data phase of a CAN FD frame is limited by the transmitter delay.

41.5.1.4.1 Description

The M_CAN's protocol unit has implemented a delay compensation mechanism to compensate the transmitter delay, thereby enabling transmission with higher bit rates during the CAN FD data phase independent of the delay of a specific CAN transceiver.

To check for bit errors during the data phase of transmitting nodes, the delayed transmit data is compared against the received data at the Secondary Sample Point SSP. If a bit error is detected, the transmitter will react on this bit error at the next following regular sample point. During arbitration phase the delay compensation is always disabled.

The transmitter delay compensation enables configurations where the data bit time is shorter than the transmitter delay, it is described in detail in the new ISO11898-1. It is enabled by setting bit DBTP[TDC].

The received bit is compared against the transmitted bit at the SSP. The SSP position is defined as the sum of the measured delay from the M_CAN's transmit output M_CAN_Tx through the transceiver to the receive input M_CAN_Rx plus the transmitter delay compensation offset as configured by TDCR[TDCO]. The transmitter delay compensation offset is used to adjust the position of the SSP inside the received bit (e.g. half of the bit time in the data phase). The position of the secondary sample point is rounded down to the next integer number of mtq.

PSR[TDCV] shows the actual transmitter delay compensation value. PSR[TDCV] is cleared when CCCR[INIT] is set and is updated at each transmission of an FD frame while DBTP[TDC] is set.

The following boundary conditions have to be considered for the transmitter delay compensation implemented in the M_CAN:

- The sum of the measured delay from M_CAN_Tx to M_CAN_Rx and the configured transmitter delay compensation offset TDCR[TDCO] has to be less than 6 bit times in the data phase.
- The sum of the measured delay from M_CAN_Tx to M_CAN_Rx and the configured transmitter delay compensation offset TDCR[TDCO] has to be less or equal 127 mtq. In case this sum exceeds 127 mtq, the maximum value of 127 mtq is used for transmitter delay compensation.
- The data phase ends at the sample point of the CRC delimiter, that stops checking of receive bits at the SSPs.

41.5.1.4.2 Transmitter delay compensation measurement

If transmitter delay compensation is enabled by programming $\text{DBTP}[\text{TDC}] = 1$, the measurement is started within each transmitted CAN FD frame at the falling edge of bit FDF to bit res. The measurement is stopped when this edge is seen at the receive input M_CAN_Rx of the transmitter. The resolution of this measurement is one mtq.

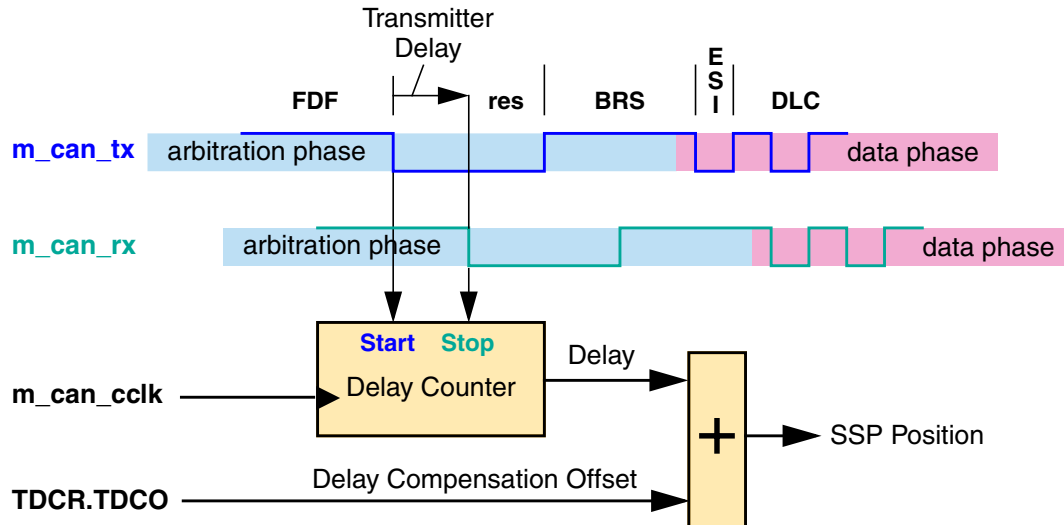


Figure 41-50. Transmitter delay measurement

To avoid that a dominant glitch inside the received FDF bit ends the delay compensation measurement before the falling edge of the received res bit, resulting in a too early SSP position, the use of a transmitter delay compensation filter window can be enabled by programming $\text{TDCR}[\text{TDCF}]$. This defines a minimum value for the SSP position. Dominant edges on M_CAN_Rx, that would result in an earlier SSP position are ignored for transmitter delay measurement. The measurement is stopped when the SSP position is at least $\text{TDCR}[\text{TDCF}]$ AND M_CAN_Rx is low.

41.5.1.5 Restricted Operation Mode

In Restricted Operation Mode the node is able to receive data and remote frames and to give acknowledge to valid frames, but it does not send data frames, remote frames, active error frames, or overload frames. In case of an error condition or overload condition, it does not send dominant bits, instead it waits for the occurrence of bus idle condition to resynchronize itself to the CAN communication. The error counters ($\text{ECR}[\text{REC}]$, $\text{ECR}[\text{TEC}]$) are frozen while Error Logging ($\text{ECR}[\text{CEL}]$) is active. The Host can set the M_CAN into Restricted Operation mode by setting bit $\text{CCCR}[\text{ASM}]$. The bit can only be set by the Host when both $\text{CCCR}[\text{CCE}]$ and $\text{CCCR}[\text{INIT}]$ are set to 1. The bit can be reset by the Host at any time.

Restricted Operation Mode is automatically entered when the Tx Handler was not able to read data from the Message RAM in time. To leave Restricted Operation Mode, the Host CPU has to reset CCCR[ASM].

The Restricted Operation Mode can be used in applications that adapt themselves to different CAN bit rates. In this case the application tests different bit rates and leaves the Restricted Operation Mode after it has received a valid frame.

NOTE

The Restricted Operation Mode must not be combined with the Loop Back Mode (internal or external).

41.5.1.6 Bus Monitoring Mode

The M_CAN is set in Bus Monitoring Mode by programming CCCR.MON to one. In Bus Monitoring Mode (see ISO11898-1, 10.12 Bus monitoring), the M_CAN is able to receive valid data frames and valid remote frames, but cannot start a transmission. In this mode, it sends only recessive bits on the CAN bus. If the M_CAN is required to send a dominant bit (ACK bit, overload flag, active error flag), the bit is rerouted internally so that the M_CAN monitors this dominant bit, although the CAN bus may remain in recessive state. In Bus Monitoring Mode, register TXBRP is held in reset state.

The Bus Monitoring Mode can be used to analyze the traffic on a CAN bus without affecting it by the transmission of dominant bits. The following figure shows the connection of signals M_CAN Tx and Rx to the M_CAN in Bus Monitoring Mode.

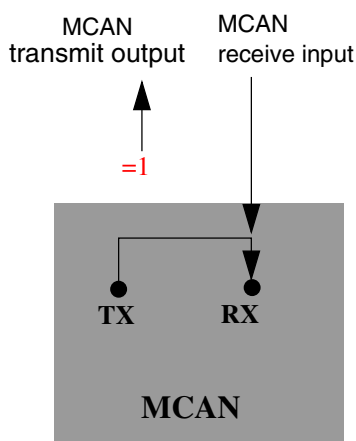


Figure 41-51. Pin Control in Bus Monitoring Mode

41.5.1.7 Disabled Automatic Retransmission

According to the CAN Specification (see ISO11898-1, 6.3.3 Recovery Management), the M_CAN provides means for automatic retransmission of frames that have lost arbitration or that have been disturbed by errors during transmission. By default automatic retransmission is enabled. To support time-triggered communication as described in ISO 11898-1, chapter 9.2, the automatic retransmission may be disabled via CCCR[DAR].

41.5.1.7.1 Frame Transmission in DAR mode

In DAR mode all transmissions are automatically cancelled after they started on the CAN bus. A Tx Buffer's Tx Request Pending bit TXBRP[TRPx] is reset after successful transmission, when a transmission has not yet been started at the point of cancellation, has been aborted due to lost arbitration, or when an error occurred during frame transmission.

- Successful transmission:
 - Corresponding Tx Buffer Transmission Occurred bit TXBTO[TOx] set
 - Corresponding Tx Buffer Cancellation Finished bit TXBCF[CFx] not set
- Successful transmission in spite of cancellation:
 - Corresponding Tx Buffer Transmission Occurred bit TXBTO[TOx] set
 - Corresponding Tx Buffer Cancellation Finished bit TXBCF[CFx] set
- Arbitration lost or frame transmission disturbed:
 - Corresponding Tx Buffer Transmission Occurred bit TXBTO[TOx] not set
 - Corresponding Tx Buffer Cancellation Finished bit TXBCF[CFx] set

In case of a successful frame transmission, and if storage of Tx events is enabled, a Tx Event FIFO element is written with Event Type ET = 10 (transmission in spite of cancellation).

41.5.1.8 Power Down (Sleep Mode)

The M_CAN can be set into power down mode controlled by CC Control Register CCCR[CSR].

When all pending transmission requests have completed, the M_CAN waits until bus idle state is detected. Then the M_CAN sets then CCCR[INIT] to one to prevent any further CAN transfers. Now the M_CAN acknowledges that it is ready for power down by

setting CCCR[CSA] to one. In this state, before the clocks are switched off, further register accesses can be made. A write access to CCCR[INIT] will have no effect. Now the module input clocks: CAN clock and Host clock may be switched off.

To leave power down mode, the application has to turn on the module clocks before resetting CC Control Register flag CCCR[CSR]. The M_CAN will acknowledge this by resetting output signal clock stop acknowledge and resetting CCCR[CSA]. Afterwards, the application can restart CAN communication by resetting bit CCCR[INIT].

41.5.1.9 Test Modes

To enable write access to register TEST, bit CCCR[TEST] has to be set to one. This allows the configuration of the test modes and test functions.

Four output functions are available for the CAN transmit pin M_CAN_Tx by programming TEST[TX]. Additionally to its default function – the serial data output – it can drive the CAN Sample Point signal to monitor the M_CAN's bit timing and it can drive constant dominant or recessive values. The actual value at pin M_CAN_Rx can be read from TEST[RX]. Both functions can be used to check the CAN bus' physical layer.

Due to the synchronization mechanism between CAN clock and Host clock domain, there may be a delay of several Host clock periods between writing to TEST[TX] until the new configuration is visible at output pin M_CAN_Tx. This applies also when reading input pin M_CAN_Rx via TEST[RX].

Note

Test modes should be used for production tests or self test only. The software control for pin M_CAN_Tx interferes with all CAN protocol functions. It is not recommended to use test modes for application.

41.5.1.9.1 External Loopback Mode

The M_CAN can be set in External Loopback Mode by programming TEST[LBCK] to one. In Loopback Mode, the M_CAN treats its own transmitted messages as received messages and stores them (if they pass acceptance filtering) into an Rx Buffer or an Rx FIFO. The following figure shows the connection of signals M_CAN_Tx and M_CAN_Rx to the M_CAN in External Loopback Mode.

This mode is provided for hardware self-test. To be independent from external stimulation, the M_CAN ignores acknowledge errors (recessive bit sampled in the acknowledge slot of a data/remote frame) in Loopback Mode. In this mode the M_CAN

performs an internal feedback from its Tx output to its Rx input. The actual value of the input pin M_CAN_Rx is disregarded by the M_CAN. The transmitted messages can be monitored at the pin M_CAN_Tx.

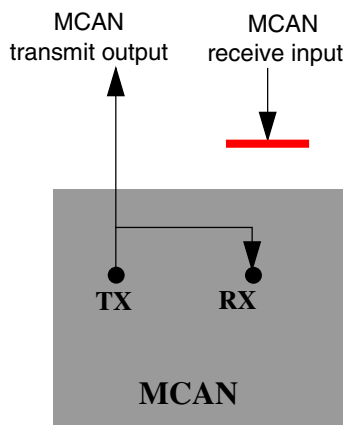


Figure 41-52. Pin Control in External Loopback Mode

41.5.1.9.2 Internal Loopback Mode

Internal Loopback Mode is entered by programming bits TEST[LBCK] and CCCR[MON] to one. This mode can be used for a "Hot Selftest", meaning the M_CAN can be tested without affecting a running CAN system connected to the pins M_CAN_Tx and M_CAN_Rx. In this mode pin M_CAN_Rx is disconnected from the M_CAN and pin M_CAN_Tx is held recessive. The following figure shows the connection of M_CAN_Tx and M_CAN_Rx to the M_CAN in case of Internal Loopback Mode.

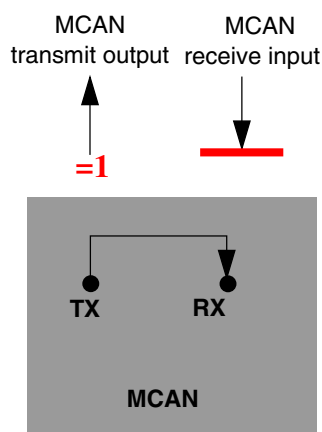


Figure 41-53. Pin Control in Internal Loopback Mode

41.5.2 Timestamp Generation

For timestamp generation the M_CAN supplies a 16-bit wrap-around counter. A prescaler TSCC[TCP] can be configured to clock the counter in multiples of CAN bit times (1...16). The counter is readable via TSCV[TSC]. A write access to register TSCV resets the counter to zero. When the timestamp counter wraps around interrupt flag IR[TSW] is set.

On start of frame reception / transmission the counter value is captured and stored into the timestamp section of an Rx Buffer / Rx FIFO (RXTS[15:0]) or Tx Event FIFO (TXTS[15:0]) element.

41.5.3 Timeout Counter

To signal timeout conditions for Rx FIFO 0, Rx FIFO 1, and the Tx Event FIFO the M_CAN supplies a 16-bit Timeout Counter. It operates as down-counter and uses the same prescaler controlled by TSCC[TCP] as the Timestamp Counter. The Timeout Counter is configured via register TOCC. The actual counter value can be read from TOCV[TOC]. The Timeout Counter can only be started while CCCR[INIT] = 0. It is stopped when CCCR[INIT] = 1, e.g. when the M_CAN enters Bus_Off state.

The operation mode is selected by TOCC[TOS]. When operating in Continuous mode, the counter starts when CCCR[INIT] is reset. A write to TOCV presets the counter to the value configured by TOCC[TOP] and continues down-counting.

When the Timeout Counter is controlled by one of the FIFOs, an empty FIFO presets the counter to the value configured by TOCC[TOP]. Down-counting is started when the first FIFO element is stored. Writing to TOCV has no effect.

When the counter reaches zero, interrupt flag IR[TOO] is set. In Continuous Mode, the counter is immediately restarted at TOCC[TOP].

Note

The clock signal for the Timeout Counter is derived from the CAN Core's sample point signal. Therefore the point in time where the Timeout Counter is decremented may vary due to the synchronization / re-synchronization mechanism of the CAN Core. If the baud rate switch feature in CAN FD is used, the timeout counter is clocked differently in arbitration and data field.

41.5.4 Rx Handling

The Rx Handler controls the acceptance filtering, the transfer of received messages to the Rx Buffers or to one of the two Rx FIFOs, as well as the Rx FIFO's Put and Get Indices.

41.5.4.1 Acceptance filtering

The M_CAN offers the possibility to configure two sets of acceptance filters, one for standard identifiers and one for extended identifiers. These filters can be assigned to Rx Buffer or to Rx FIFO 0, 1. For acceptance filtering each list of filters is executed from element #0 until the first matching element. Acceptance filtering stops at the first matching element. The following filter elements are not evaluated for this message.

The main features are:

- Each filter element can be configured as
 - Range filter (from - to)
 - Filter for one or two dedicated IDs
 - Classic bit mask filter
- Each filter element is configurable for acceptance or rejection filtering
- Each filter element can be enabled / disabled individually
- Filters are checked sequentially, execution stops with the first matching filter element

Related configuration registers are:

- Global Filter Configuration (GFC)
- Standard ID Filter Configuration (SIDFC)
- Extended ID Filter Configuration (XIDFC)
- Extended ID AND Mask (XIDAM)

Depending on the configuration of the filter element (SFEC/EFEC) a match triggers one of the following actions:

- Store received frame in FIFO 0 or FIFO 1
- Store received frame in Rx Buffer
- Store received frame in Rx Buffer and generate pulse at filter event pin

- Reject received frame
- Set High Priority Message interrupt flag IR.HPM
- Set High Priority Message interrupt flag IR.HPM and store received frame in FIFO 0 or FIFO 1

Acceptance filtering is started after the complete identifier has been received. After acceptance filtering has completed, and if a matching Rx Buffer or Rx FIFO has been found, the Message Handler starts writing the received message data in portions of 32 bit to the matching Rx Buffer or Rx FIFO. If the CAN protocol controller has detected an error condition (e.g. CRC error), this message is discarded with the following impact on the affected Rx Buffer or Rx FIFO:

Rx Buffer

New Data flag of matching Rx Buffer is not set, but Rx Buffer (partly) overwritten with received data. For error type see PSR[LEC] respectively PSR[DLEC].

Rx FIFO

Put index of matching Rx FIFO is not updated, but related Rx FIFO element (partly) overwritten with received data. For error type see PSR[LEC] respectively PSR[DLEC]. In case the matching Rx FIFO is operated in overwrite mode, the boundary conditions described in "Rx FIFO Overwrite Mode" section have to be considered.

Note

When an accepted message is written to one of the two Rx FIFOs, or into an Rx Buffer, the unmodified received identifier is stored independent of the filter(s) used. The result of the acceptance filter process is strongly depending on the sequence of configured filter elements.

41.5.4.1.1 Range Filter

The filter matches for all received frames with Message IDs in the range defined by SF1ID/SF2ID resp. EF1ID/EF2ID.

There are two possibilities when range filtering is used together with extended frames:

- EFT = "00": The Message ID of received frames is ANDed with the Extended ID AND Mask (XIDAM) before the range filter is applied
- EFT = "11": The Extended ID AND Mask (XIDAM) is not used for range filtering

41.5.4.1.2 Filter for Specific IDs

A filter element can be configured to filter for one or two specific Message IDs. To filter for one specific Message ID, the filter element has to be configured with SF1ID = SF2ID resp. EF1ID = EF2ID.

41.5.4.1.3 Classic Bit Mask Filter

Classic bit mask filtering is intended to filter groups of Message IDs by masking single bits of a received Message ID. With classic bit mask filtering SF1ID/EF1ID is used as Message ID filter, while SF2ID/EF2ID is used as filter mask.

A zero bit at the filter mask will mask out the corresponding bit position of the configured ID filter, e.g. the value of the received Message ID at that bit position is not relevant for acceptance filtering. Only those bits of the received Message ID where the corresponding mask bits are one are relevant for acceptance filtering.

In case all mask bits are one, a match occurs only when the received Message ID and the Message ID filter are identical. If all mask bits are zero, all Message IDs match.

41.5.4.1.4 Standard Message ID Filtering

The following figure shows the flow for standard Message ID (11-bit Identifier) filtering. The Standard Message ID Filter element is described in [Standard Message ID Filter Element](#).

Controlled by the Global Filter Configuration GFC and the Standard ID Filter Configuration SIDFC Message ID, Remote Transmission Request bit (RTR), and the Identifier Extension bit (IDE) of received frames are compared against the list of configured filter elements.

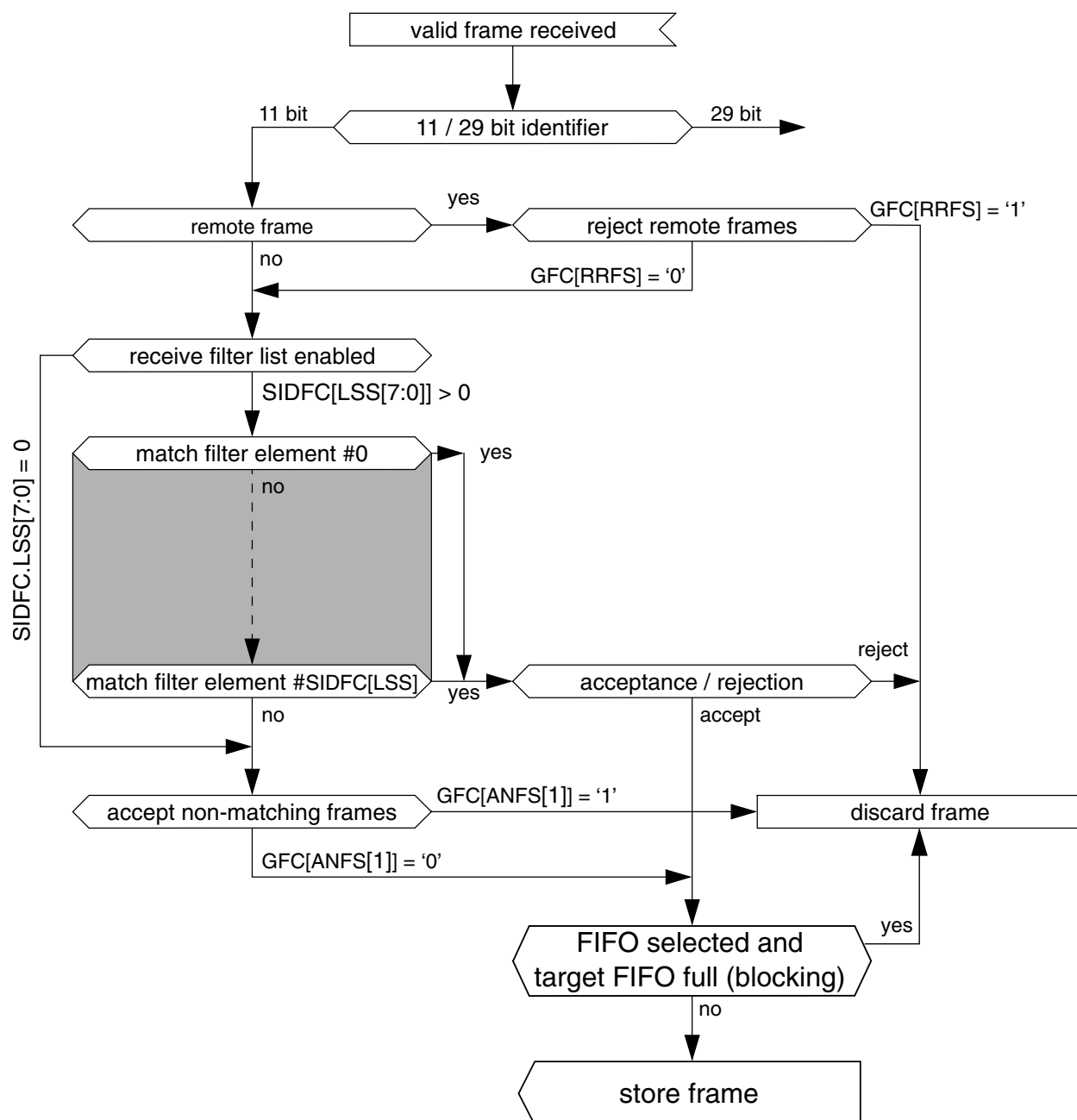


Figure 41-54. Standard Message ID Filter Path

41.5.4.1.5 Extended Message ID Filtering

The figure below shows the flow for extended Message ID (29-bit Identifier) filtering. The Extended Message ID Filter element is described in [Extended Message ID Filter Element](#).

Controlled by the Global Filter Configuration GFC and the Extended ID Filter Configuration XIDFC Message ID, Remote Transmission Request bit (RTR), and the Identifier Extension bit (IDE) of received frames are compared against the list of configured filter elements.

The Extended ID AND Mask XIDAM is ANDed with the received identifier before the filter list is executed.

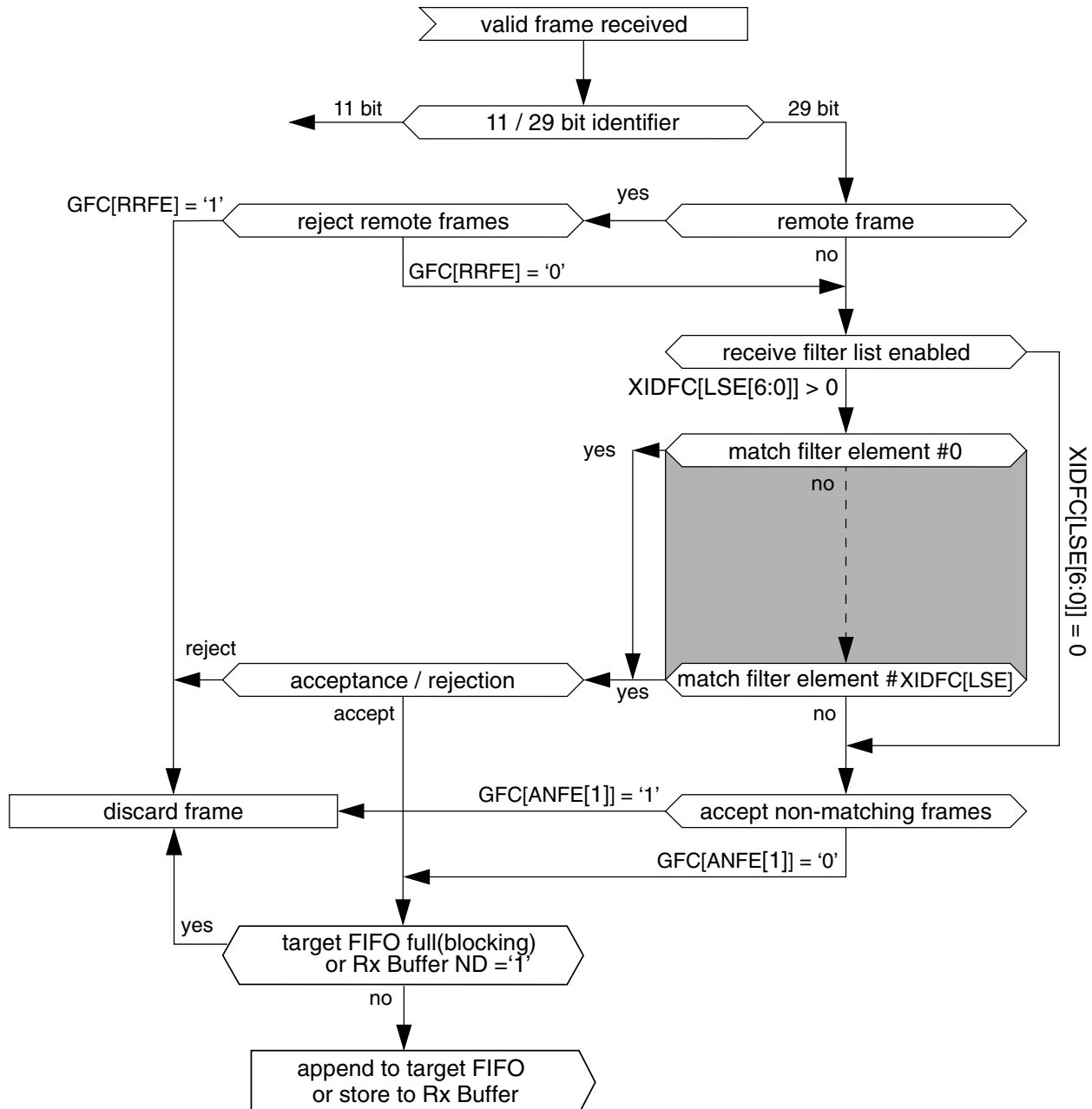


Figure 41-55. Extended Message ID Filter Path

41.5.4.1.6 Rx FIFOs

Rx FIFO 0 and Rx FIFO 1 can be configured to hold up to 64 elements each. Configuration of the two Rx FIFOs is done via registers RXF0C and RXF1C.

Received messages that passed acceptance filtering are transferred to the Rx FIFO as configured by the matching filter element. For a description of the filter mechanisms available for Rx FIFO 0 and Rx FIFO 1 see [Acceptance filtering](#). The Rx FIFO element is described in [Rx Buffer and FIFO Element](#).

To avoid an Rx FIFO overflow, the Rx FIFO watermark can be used. When the Rx FIFO fill level reaches the Rx FIFO watermark configured by RXFnC[FnWM], interrupt flag IR[RFnW] is set. When the Rx FIFO Put Index reaches the Rx FIFO Get Index an Rx FIFO Full condition is signalled by RXFnS[FnF]. In addition interrupt flag IR[RFnF] is set.

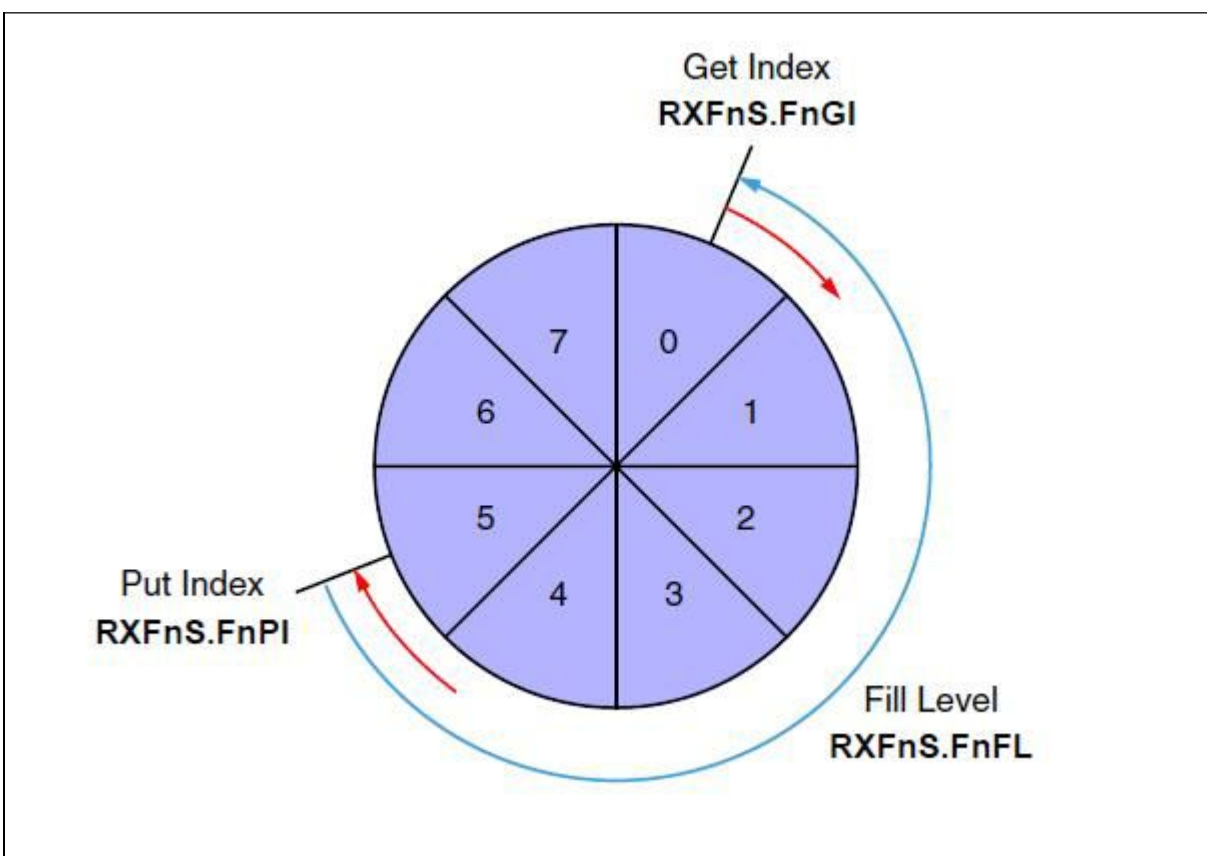


Figure 41-56. Rx FIFO Status

When reading from an Rx FIFO, Rx FIFO Get Index RXFnS[FnGI] x FIFO Element Size has to be added to the corresponding Rx FIFO start address RXFnC[FnSA].

Table 41-60. Rx Buffer / FIFO Element Size

RXESC.RBDS[2:0] RXESC.FnDS[2:0]	Data Field [bytes]	FIFO Element Size [RAM words]
000	8	4
001	12	5
010	16	6
011	20	7
100	24	8
101	32	10
110	48	14
111	64	18

41.5.4.1.6.1 Rx FIFO Blocking Mode

The Rx FIFO blocking mode is configured by $RXFnC[FnOM] = 0$. This is the default operation mode for the Rx FIFOs.

When an Rx FIFO full condition is reached ($RXFnS[FnPI] = RXFnS[FnGI]$), no further messages are written to the corresponding Rx FIFO until at least one message has been read out and the Rx FIFO Get Index has been incremented. An Rx FIFO full condition is signalled by $RXFnS[FnF] = 1$. In addition interrupt flag $IR[RFnF]$ is set.

In case a message is received while the corresponding Rx FIFO is full, this message is discarded and the message lost condition is signalled by $RXFnS[RFnL] = 1$. In addition interrupt flag $IR[RFnL]$ is set.

41.5.4.1.6.2 Rx FIFO Overwrite Mode

The Rx FIFO overwrite mode is configured by $RXFnC[FnOM] = 1$.

When an Rx FIFO full condition ($RXFnS[FnPI] = RXFnS[FnGI]$) is signalled by $RXFnS[FnF] = 1$, the next message accepted for the FIFO will overwrite the oldest FIFO message. Put and get index are both incremented by one.

When an Rx FIFO is operated in overwrite mode and an Rx FIFO full condition is signalled, reading of the Rx FIFO elements should start at least at get index + 1. The reason for that is, that it might happen, that a received message is written to the Message RAM (put index) while the CPU is reading from the Message RAM (get index). In this case inconsistent data may be read from the respective Rx FIFO element. Adding an offset to the get index when reading from the Rx FIFO avoids this problem. The offset depends on how fast the CPU accesses the Rx FIFO. The following figure shows an offset of two with respect to the get index when reading the Rx FIFO. In this case the two messages stored in element 1 and 2 are lost.

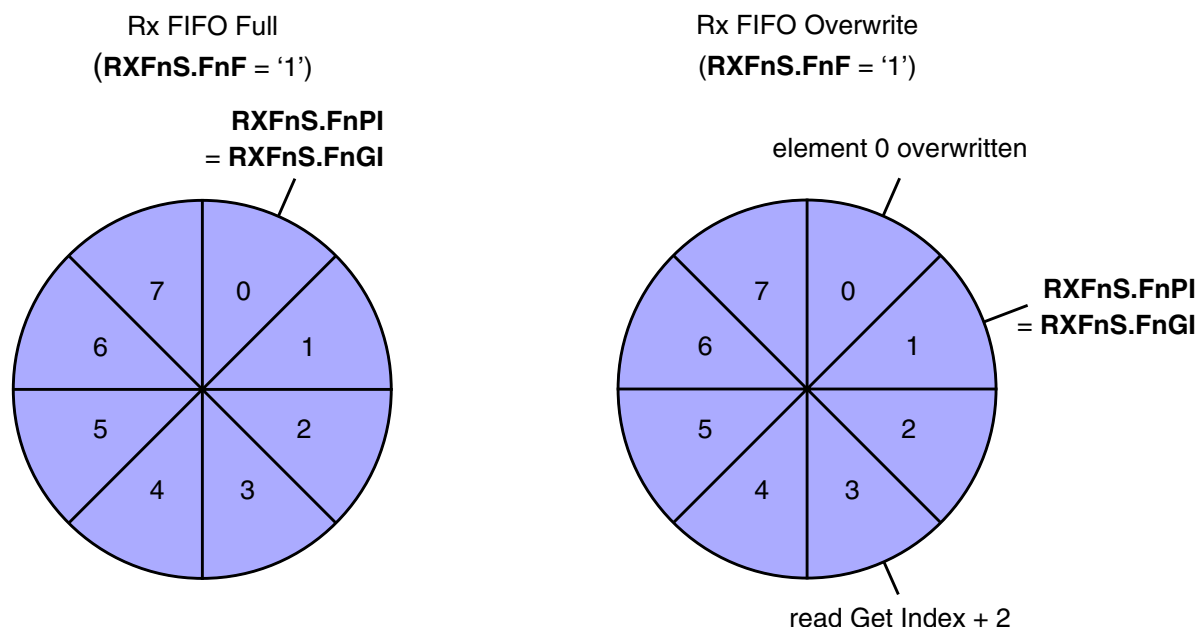


Figure 41-57. Rx FIFO Overflow Handling

After reading from the Rx FIFO, the number of the last element read has to be written to the Rx FIFO Acknowledge Index RXFnA[FnA]. This increments the get index to that element number. In case the put index has not been incremented to this Rx FIFO element, the Rx FIFO full condition is reset (RXFnS[FnF] = 0).

41.5.4.1.7 Dedicated Rx Buffers

The M_CAN supports up to 64 dedicated Rx Buffers. The start address of the dedicated Rx Buffer section is configured via RXBC[RBSA].

For each Rx Buffer a Standard or Extended Message ID Filter Element with SFEC / EFEC = 111 and SFID2 / EFID2[10:9] = 00 has to be configured (see [Standard Message ID Filter Element](#) and [Extended Message ID Filter Element](#))

After a received message has been accepted by a filter element, the message is stored into the Rx Buffer in the Message RAM referenced by the filter element. The format is the same as for an Rx FIFO element. In addition the flag IR[DRX] (Message stored in Dedicated Rx Buffer) in the interrupt register is set.

Table 41-61. Example Filter Configuration for Rx buffers

Filter Element	SFID1[10:0] EFID1[28:0]	SFID2[10:9] EFID2[10:9]	SFID2[5:0] EFID2[5:0]
0	ID message 1	00	00 0000
1	ID message 2	00	00 0001
2	ID message 3	00	00 0010

After the last word of a matching received message has been written to the Message RAM, the respective New Data flag in register NDAT1,2 is set. As long as the New Data flag is set, the respective Rx Buffer is locked against updates from received matching frames. The New Data flags have to be reset by the Host by writing a 1 to the respective bit position.

While an Rx Buffer's New Data flag is set, a Message ID Filter Element referencing this specific Rx Buffer will not match, causing the acceptance filtering to continue. Following Message ID Filter Elements may cause the received message to be stored into another Rx Buffer, or into an Rx FIFO, or the message may be rejected, depending on filter configuration

41.5.4.1.7.1 Rx Buffer Handling

- Reset interrupt flag IR[DRX]
- Read New Data registers
- Read messages from Message RAM
- Reset New Data flags of processed messages

41.5.4.1.8 Debug on CAN Support

Debug messages are stored into Rx Buffers. For debug handling three consecutive Rx buffers (e.g. #61, #62, #63) have to be used for storage of debug messages A, B, and C. The format is the same as for an Rx Buffer or an Rx FIFO element.

Advantage: Fixed start address for the DMA transfers (relative to RXBC[RBSA]), no additional configuration required.

For filtering of debug messages Standard / Extended Filter Elements with SFEC / EFEC = 111 have to be set up. Messages matching these filter elements are stored into the Rx Buffers addressed by SFID2 / EFID2[5:0].

After message C has been stored, the DMA request output m_can_dma_req is activated and the three messages can be read from the Message RAM under DMA control. The RAM words holding the debug messages will not be changed by the M_CAN while m_can_dma_req is activated. The behaviour is similar to that of an Rx Buffers with its New Data flag set.

After the DMA has completed the DMA unit sets m_can_dma_ack. This resets m_can_dma_req. Now the M_CAN is prepared to receive the next set of debug messages.

41.5.4.1.8.1 Filtering for Debug Messages

Filtering for debug messages is done by configuring one Standard / Extended Message ID Filter Element for each of the three debug messages. To enable a filter element to filter for debug messages SFEC / EFEC has to be programmed to 111. In this case fields SFID1 / SFID2 and EFID1 / EFID2 have a different meaning (see [Standard Message ID Filter Element](#) and [Extended Message ID Filter Element](#)). While SFID2 / EFID2[10:9] controls the debug message handling state machine, SFID2 / EFID2[5:0] controls the location for storage of a received debug message.

When a debug message is stored, neither the respective New Data flag nor IR[DRX] are set. The reception of debug messages can be monitored via RXF1S[DMS].

Table 41-62. Example Filter Configuration for Debug Messages

Filter Element	SFID1[10:0] EFID1[28:0]	SFID2[10:9] EFID2[10:9]	SFID2[5:0] EFID2[5:0]
0	ID debug message A	01	11 1101
1	ID debug message B	10	11 1110
2	ID debug message C	11	11 1111

41.5.4.1.8.2 Debug Message Handling

The debug message handling state machine assures that debug messages are stored to three consecutive Rx Buffers in correct order. In case of missing messages the process is restarted. The DMA request is activated only when all three debug messages A, B, C have been received in correct order.

The status of the debug message handling state machine is signaled via RXF1S[DMS].

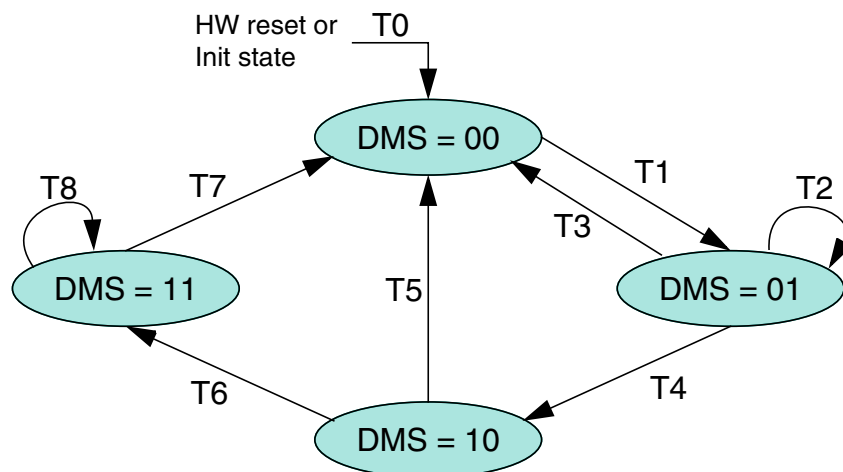


Figure 41-58. Debug Message Handling State Machine

T0: reset m_can_dma_req output, enable reception of debug messages A, B, and C

T1: reception of debug message A

T2: reception of debug message A

T3: reception of debug message C

T4: reception of debug message B

T5: reception of debug messages A, B

T6: reception of debug message C

T7: DMA transfer completed

T8: reception of debug message A,B,C (message rejected)

41.5.5 Tx Handling

The Tx Handler handles transmission requests for the dedicated Tx Buffers, the Tx FIFO, and the Tx Queue. It controls the transfer of transmit messages to the CAN Core, the Put and Get Indices, and the Tx Event FIFO. Up to 32 Tx Buffers can be set up for message transmission. The CAN mode for transmission (Classic CAN or CAN FD) can be configured separately for each Tx Buffer element. The Tx Buffer element is described in [Tx Buffer Element](#). The table below describes the possible configurations for frame transmission.

Table 41-63. Possible configurations for frame transmission

CCCR		Tx Buffer Element		Frame Transmission
BRSE	FDOE	FDF	BRS	
ignored	0	ignored	ignored	Classic CAN
0	1	0	ignored	Classic CAN
0	1	1	ignored	FD without bit rate switching
1	1	0	ignored	Classic CAN
1	1	1	0	FD without bit rate switching
1	1	1	1	FD with bit rate switching

Note

AUTOSAR requires at least three Tx Queue Buffers and support of transmit cancellation.

The Tx Handler starts a Tx scan to check for the highest priority pending Tx request (Tx Buffer with lowest Message ID) when the Tx Buffer Request Pending register TXBRP is updated, or when a transmission has been started.

41.5.5.1 Transmit Pause

The transmit pause feature is intended for use in CAN systems where the CAN message identifiers are (permanently) specified to specific values and cannot easily be changed. These message identifiers may have a higher CAN arbitration priority than other defined messages, while in a specific application their relative arbitration priority should be inverse. This may lead to a case where one ECU sends a burst of CAN messages that cause another ECU's CAN messages to be delayed because that other messages have a lower CAN arbitration priority.

If, for example, CAN ECU-1 has the transmit pause feature enabled and is requested by its application software to transmit four messages, it will, after the first successful message transmission, wait for two CAN bit times of bus idle before it is allowed to start the next requested message. If there are other ECUs with pending messages, those messages are started in the idle time, they would not need to arbitrate with the next message of ECU-1. After having received a message, ECU-1 is allowed to start its next transmission as soon as the received message releases the CAN bus.

The transmit pause feature is controlled by bit CCCR.TXP. If the bit is set, the M_CAN will, each time it has successfully transmitted a message, pause for two CAN bit times before starting the next transmission. This enables other CAN nodes in the network to transmit messages even if their messages have lower prior identifiers. Default is transmit pause disabled (CCCR[TXP] = 0).

This feature looses up burst transmissions coming from a single node and it protects against “babbling idiot” scenarios where the application program erroneously requests too many transmissions.

41.5.5.2 Dedicated Tx Buffers

Dedicated Tx Buffers are intended for message transmission under complete control of the Host CPU. Each Dedicated Tx Buffer is configured with a specific Message ID. In case that multiple Tx Buffers are configured with the same Message ID, the Tx Buffer with the lowest buffer number is transmitted first.

If the data section has been updated, a transmission is requested by an Add Request via TXBAR[ARn]. The requested messages arbitrate internally with messages from an optional Tx FIFO or Tx Queue and externally with messages on the CAN bus, and are sent out according to their Message ID.

A Dedicated Tx Buffer allocates Element Size 32-bit words in the Message RAM (see the following table). Therefore the start address of a Dedicated Tx Buffer in the Message RAM is calculated by adding transmit buffer index (0...31) x Element Size to the Tx Buffer Start Address TXBC[TBSA].

Table 41-64. Tx Buffer / FIFO / Queue Element Size

TXESC.TBDS[2:0]	Data Field [bytes]	Element Size [RAM words]
000	8	4
001	12	5
010	16	6
011	20	7
100	24	8
101	32	10
110	48	14
111	64	18

41.5.5.3 Tx FIFO

Tx FIFO operation is configured by programming TXBC[TFQM] to 0. Messages stored in the Tx FIFO are transmitted starting with the message referenced by the Get Index TXFQS[TFGI]. After each transmission the Get Index is incremented cyclically until the Tx FIFO is empty. The Tx FIFO enables transmission of messages with the same Message ID from different Tx Buffers in the order these messages have been written to the Tx FIFO. The M_CAN calculates the Tx FIFO Free Level TXFQS[TFFL] as difference between Get and Put Index. It indicates the number of available (free) Tx FIFO elements.

New transmit messages have to be written to the Tx FIFO starting with the Tx Buffer referenced by the Put Index TXFQS[TFQPI]. An Add Request increments the Put Index to the next free Tx FIFO element. When the Put Index reaches the Get Index, Tx FIFO Full (TXFQS[TFQF] = 1) is signalled. In this case no further messages should be written to the Tx FIFO until the next message has been transmitted and the Get Index has been incremented.

When a single message is added to the Tx FIFO, the transmission is requested by writing a 1 to the TXBAR bit related to the Tx Buffer referenced by the Tx FIFO's Put Index.

When multiple (n) messages are added to the Tx FIFO, they are written to n consecutive Tx Buffers starting with the Put Index. The transmissions are then requested via TXBAR. The Put Index is then cyclically incremented by n. The number of requested Tx buffers should not exceed the number of free Tx Buffers as indicated by the Tx FIFO Free Level.

When a transmission request for the Tx Buffer referenced by the Get Index is cancelled, the Get Index is incremented to the next Tx Buffer with pending transmission request and the Tx FIFO Free Level is recalculated. When transmission cancellation is applied to any other Tx Buffer, the Get Index and the FIFO Free Level remain unchanged.

A Tx FIFO element allocates Element Size 32-bit words in the Message RAM (See [Table 41-64](#)). Therefore the start address of the next available (free) Tx FIFO Buffer is calculated by adding Tx FIFO/Queue Put Index TXFQS.TFQPI (0...31) x Element Size to the Tx Buffer Start Address TXBC[TBSA].

41.5.5.4 Tx Queue

Tx Queue operation is configured by programming TXBC[TFQM] to 1. Messages stored in the Tx Queue are transmitted starting with the message with the lowest Message ID (highest priority). In case that multiple Queue Buffers are configured with the same Message ID, the Queue Buffer with the lowest buffer number is transmitted first.

New messages have to be written to the Tx Buffer referenced by the Put Index TXFQS[TFQPI]. An Add Request cyclically increments the Put Index to the next free Tx Buffer. In case that the Tx Queue is full (TXFQS[TFQF] = 1), the Put Index is not valid and no further message should be written to the Tx Queue until at least one of the requested messages has been sent out or a pending transmission request has been cancelled.

The application may use register TXBRP instead of the Put Index and may place messages to any Tx Buffer without pending transmission request.

A Tx Queue Buffer allocates Element Size 32-bit words in the Message RAM (see [Table 41-64](#)). Therefore the start address of the next available (free) Tx Queue Buffer is calculated by adding Tx FIFO/Queue Put Index TXFQS[TFQPI] (0...31) x Element Size to the Tx Buffer Start Address TXBC[TBSA].

41.5.5.5 Mixed Dedicated Tx Buffers / Tx FIFO

In this case the Tx Buffers section in the Message RAM is subdivided into a set of Dedicated Tx Buffers and a Tx FIFO. The number of Dedicated Tx Buffers is configured by TXBC[NDTB]. The number of Tx Buffers assigned to the Tx FIFO is configured by TXBC[TFQS]. In case TXBC[TFQS] is programmed to zero, only Dedicated Tx Buffers are used.

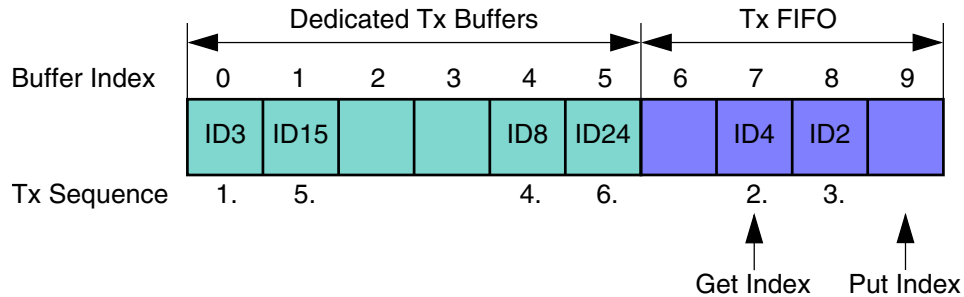


Figure 41-59. Example of mixed Configuration Dedicated Tx Buffers / Tx FIFO

Tx prioritization:

- Scan Dedicated Tx Buffers and oldest pending Tx FIFO Buffer (referenced by TXFS[TFGI])
- Buffer with lowest Message ID gets highest priority and is transmitted next

41.5.5.6 Mixed Dedicated Tx Buffers / Tx Queue

In this case the Tx Buffers section in the Message RAM is subdivided into a set of Dedicated Tx Buffers and a Tx Queue. The number of Dedicated Tx Buffers is configured by TXBC.NDTB. The number of Tx Queue Buffers is configured by TXBC.TFQS. In case TXBC.TFQS is programmed to zero, only Dedicated Tx Buffers are used.

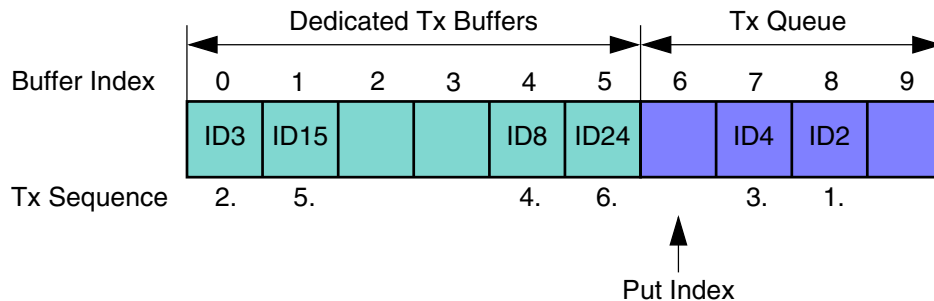


Figure 41-60. Example of mixed Configuration Dedicated Tx Buffers / Tx Queue

Tx prioritization:

- Scan all Tx Buffers with activated transmission request
- Tx Buffer with lowest Message ID gets highest priority and is transmitted next

41.5.5.7 Transmit Cancellation

The M_CAN supports transmit cancellation. This feature is especially intended for gateway applications and AUTOSAR based applications. To cancel a requested transmission from a dedicated Tx Buffer or a Tx Queue Buffer the Host has to write a 1 to the corresponding bit position (= number of Tx Buffer) of register TXBCR. Transmit cancellation is not intended for Tx FIFO operation.

Successful cancellation is signalled by setting the corresponding bit of register TXBCF to 1.

In case a transmit cancellation is requested while a transmission from a Tx Buffer is already ongoing, the corresponding TXBRP bit remains set as long as the transmission is in progress. If the transmission was successful, the corresponding TXBTO and TXBCF bits are set. If the transmission was not successful, it is not repeated and only the corresponding TXBCF bit is set.

Note

In case a pending transmission is cancelled immediately before this transmission could have been started, there follows a short time window where no transmission is started even if another message is also pending in this node. This may enable another node to transmit a message which may have a lower priority than the second message in this node.

41.5.5.8 Tx Event Handling

To support Tx event handling the M_CAN has implemented a Tx Event FIFO. After the M_CAN has transmitted a message on the CAN bus, Message ID and timestamp are stored in a Tx Event FIFO element. To link a Tx event to a Tx Event FIFO element, the Message Marker from the transmitted Tx Buffer is copied into the Tx Event FIFO element.

The Tx Event FIFO can be configured to a maximum of 32 elements. The Tx Event FIFO element is described in [Tx Event FIFO Element](#).

The purpose of the Tx Event FIFO is to decouple handling transmit status information from transmit message handling i.e. a Tx Buffer holds only the message to be transmitted, while the transmit status is stored separately in the Tx Event FIFO. This has the advantage, especially when operating a dynamically managed transmit queue, that a Tx Buffer can be used for a new message immediately after successful transmission. There is no need to save transmit status information from a Tx Buffer before overwriting that Tx Buffer.

When a Tx Event FIFO full condition is signalled by IR[TEFF], no further elements are written to the Tx Event FIFO until at least one element has been read out and the Tx Event FIFO Get Index has been incremented. In case a Tx event occurs while the Tx Event FIFO is full, this event is discarded and interrupt flag IR[TEFL] is set.

To avoid a Tx Event FIFO overflow, the Tx Event FIFO watermark can be used. When the Tx Event FIFO fill level reaches the Tx Event FIFO watermark configured by TXEFC[EFWM], interrupt flag IR[TEFW] is set.

When reading from the Tx Event FIFO, two times the Tx Event FIFO Get Index TXEFS[EFGI] has to be added to the Tx Event FIFO start address TXEFC[EFSA].

41.5.6 FIFO Acknowledge Handling

The Get Indices of Rx FIFO 0, Rx FIFO 1, and the Tx Event FIFO are controlled by writing to the corresponding FIFO Acknowledge Index (see the registers RXF0A, RXF1A, and TXEFA). Writing to the FIFO Acknowledge Index will set the FIFO Get Index to the FIFO Acknowledge Index plus one and thereby updates the FIFO Fill Level. There are two use cases:

- When only a single element has been read from the FIFO (the one being pointed to by the Get Index), this Get Index value is written to the FIFO Acknowledge Index.
- When a sequence of elements has been read from the FIFO, it is sufficient to write the FIFO Acknowledge Index only once at the end of that read sequence (value: Index of the last element read), to update the FIFO's Get Index.

Due to the fact that the CPU has free access to the M_CAN's Message RAM, special care has to be taken when reading FIFO elements in an arbitrary order (Get Index not considered). This might be useful when reading a High Priority Message from one of the two Rx FIFOs. In this case the FIFO's Acknowledge Index should not be written because this would set the Get Index to a wrong position and also alters the FIFO's Fill Level. In this case some of the older FIFO elements would be lost.

Note

The application has to ensure that a valid value is written to the FIFO Acknowledge Index. The M_CAN does not check for erroneous values.

Chapter 42

Deserial Serial Peripheral Interface (DSPI)

42.1 Chip-specific DSPI information

This chip implements five instances of the DSPI module. Some feature details vary among the instances.

The following table summarizes the feature differences. The table does not list feature details that all instances share.

Table 42-1. DSPI instance feature differences

Feature	DSPI_A	DSPI_B	DSPI_C	DSPI_D	DSPI_E
Serialization mechanisms defined in the SIU chapter	Yes	Yes	Yes	Yes	No ¹
Serial Data In (SIN) signals	Yes	Yes	Yes	Yes	No
PISRs and bit muxes are implemented	Yes	Yes	Yes	Yes	No ²
Hardware Trigger (HT)	Yes	Yes	Yes	Yes	No
Clock and data signals (SCK and SOUT) are available on LVDS pairs to support microsecond bus functionality	No	No	Yes	Yes	No
Output serialization bits used	None	[15:0]	[15:0]	[15:4,1:0]	None

1. DSPI_E does not support serialization, except what is provided by the ASDR register.

2. The registers PSIR0, PSIR1, PSIR2, and PISR3 do not exist for DSPI_E.

42.1.1 DSPI and eMIOS direct connections

The following table shows direct connections between eMIOS and DSPI module instances.

Table 42-2. DSPI and eMIOS direct connections

Source	Destination	Description
DSPI_B	eMIOS_0	Connection from DSPI_B deserial data out bit 14 to eMIOS_0 input channel 5
DSPI_B	eMIOS_0	Connection from DSPI_B deserial data out bit 15 to eMIOS_0 input channel 4
DSPI_D	eMIOS_1	Connection from DSPI_D deserial data out bit 14 to eMIOS_1 input channel 7
DSPI_D	eMIOS_1	Connection from DSPI_D deserial data out bit 15 to eMIOS_1 input channel 6

42.1.2 DSPI connections with SIU, eTPU, and eMIOS modules

Some DSPI deserialized outputs can be used to trigger external interrupt requests via the External Interrupt Input Multiplexing sub-block in the SIU. See [External IRQ Input Select Register \(SIU_EIISR\)](#) for details.

DSPI_A, DSPI_B, DSPI_C, and DSPI_D connect via SIU registers to channels of eTPU and eMIOS modules as well as GPO data registers. See [Pin serialization for DSPI](#) for more information. The following table lists the applicable SIU registers for each DSPI instance.

Table 42-3. SIU registers for DSPI pin serialization

DSPI instance	SIU registers
DSPI_A	Mask-Output High Register (SIU_DSPIAH) Mask-Output Low Register (SIU_DSPIAL) Serialized Output Signal Selection Register for DSPI_A (SIU_ETPUBA) Serialized Output Signal Selection Register for DSPI_A (SIU_EMIOSA) Serialized Output Signal Selection Register for DSPI_A (SIU_DSPIAHLA)
DSPI_B	Mask-Output High Register (SIU_DSPIBH) Mask-Output Low Register (SIU_DSPIBL) Serialized Output Signal Selection Register for DSPI_B (SIU_ETPUAB) Serialized Output Signal Selection Register for DSPI_B (SIU_EMIOSB) Serialized Output Signal Selection Register for DSPI_B (SIU_DSPIBHLB)
DSPI_C	Mask-Output High Register (SIU_DSPICH) Mask-Output Low Register (SIU_DSPICL) Serialized Output Signal Selection Register for DSPI_C (SIU_ETPUAC) Serialized Output Signal Selection Register for DSPI_C (SIU_EMIOSC) Serialized Output Signal Selection Register for DSPI_C (SIU_DSPICHLA) Serialized Output Signal Selection Register for DSPI_C (SIU_ETPUBC)

Table continues on the next page...

Table 42-3. SIU registers for DSPI pin serialization (continued)

DSPI instance	SIU registers
DSPI_D	Mask-Output High Register (SIU_DSPIDH) Mask-Output Low Register (SIU_DSPIDL) Serialized Output Signal Selection Register for DSPI_D (SIU_ETPUBD) Serialized Output Signal Selection Register for DSPI_D (SIU_EMIOSD) Serialized Output Signal Selection Register for DSPI_D (SIU_DSPIDHLD)

42.2 Introduction

The deserial serial peripheral interface (DSPI) module provides a synchronous serial bus for communication between a chip and an external peripheral device. The module supports chip pin count reduction through serialization and deserialization of chip internal signals transmitted over the DSPI serial link.

42.2.1 Block Diagram

The block diagram of this module is as follows:

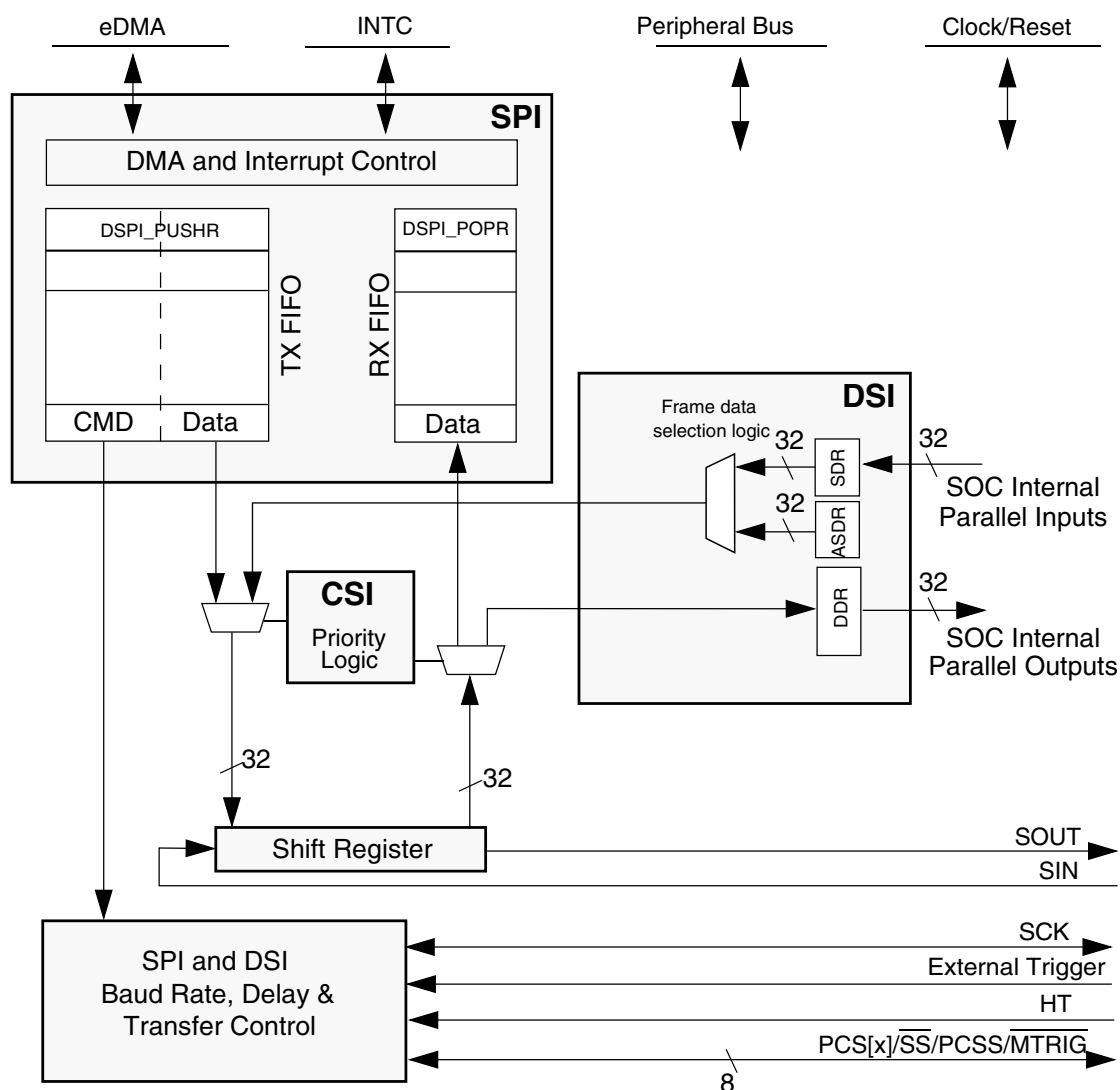


Figure 42-1. DSPI Block Diagram

42.2.2 Features

The module supports the following features:

- Full-duplex, three-wire synchronous transfers
- Master mode
- Slave mode
- Data streaming operation in Slave mode with continuous slave selection
- Buffered transmit operation using the transmit first in first out (TX FIFO) with depth of 4 entries

- Buffered receive operation using the receive FIFO (RX FIFO) with depth of 4 entries
- Asynchronous clocking scheme for Register and Protocol Interfaces
- TX and RX FIFOs can be disabled individually for low-latency updates to SPI queues
- Visibility into TX and RX FIFOs for ease of debugging
- Programmable transfer attributes on a per-frame basis:
 - eight transfer attribute registers
 - Serial clock (SCK) with programmable polarity and phase
 - Various programmable delays
 - Programmable serial frame size: 4 to 32
 - SPI frames longer than 16 bits can be supported using the continuous selection format.
 - Continuously held chip select capability
 - Parity control
- 6 peripheral chip selects (PCSeS), expandable to 64 with external demultiplexer
- Deglitching support for up to 32 peripheral chip selects (PCSeS) with external demultiplexer
- DMA support for adding entries to TX FIFO and removing entries from RX FIFO:
 - TX FIFO is not full (TFFF)
 - RX FIFO is not empty (RFDF)
- Interrupt conditions:
 - End of Queue reached (EOQF)
 - TX FIFO is not full (TFFF)
 - Transfer of current frame complete (TCF)
 - Attempt to transmit with an empty Transmit FIFO (TFUF)
 - RX FIFO is not empty (RFDF)
 - Frame received while Receive FIFO is full (RFOF)
 - SPI Parity Error (SPEF)

- Modified SPI transfer formats for communication with slower peripheral devices
- Power-saving architectural features:
 - Support for Stop mode

The DSPI also supports pin reduction through serialization and deserialization if enabled for the module.

- Two sources of serialized data:
 - The module memory-mapped register
 - Parallel Input signals
 - Programmable selection of source data on bit basis
- Deserialized data is provided as Parallel Output signals and as bits in a memory-mapped register
- Interrupt conditions:
 - Deserialized data matches pre-programmed pattern (DDIF)
 - DSI Parity Error (DPEF)
- Transfer initiation conditions:
 - Continuous
 - Edge sensitive hardware trigger
 - Change in data
- Pin serialization/deserialization with interleaved SPI frames for control and diagnostics
- Continuous serial communications clock

DSPI supports a combined SPI and DSI modes of operation known as the Combined Serial Interface (CSI). It also supports the downstream Micro Second Channel in the Timed Serial Bus (TSB) configuration.

TSB mode has the following features:

- Transmission of frames is done at frame boundaries
- Frames from SPI and DSI are identifiable by a bit transmitted at the start of each frame
- Separate interrupts for frame completion from SPI and DSI

42.2.3 Interface configurations

42.2.3.1 SPI configuration

The Serial Peripheral Interface (SPI) configuration allows the module to send and receive serial data. This configuration allows the module to operate as a basic SPI block with internal FIFOs supporting external queue operation. Transmitted data and received data reside in separate FIFOs. The host CPU or a DMA controller read the received data from the Receive FIFO and write transmit data to the Transmit FIFO.

For queued operations, the SPI queues can reside in system RAM, external to the module. Data transfers between the queues and the module FIFOs are accomplished by a DMA controller or host CPU. The following figure shows a system example with DMA, DSPI, and external queues in system RAM.

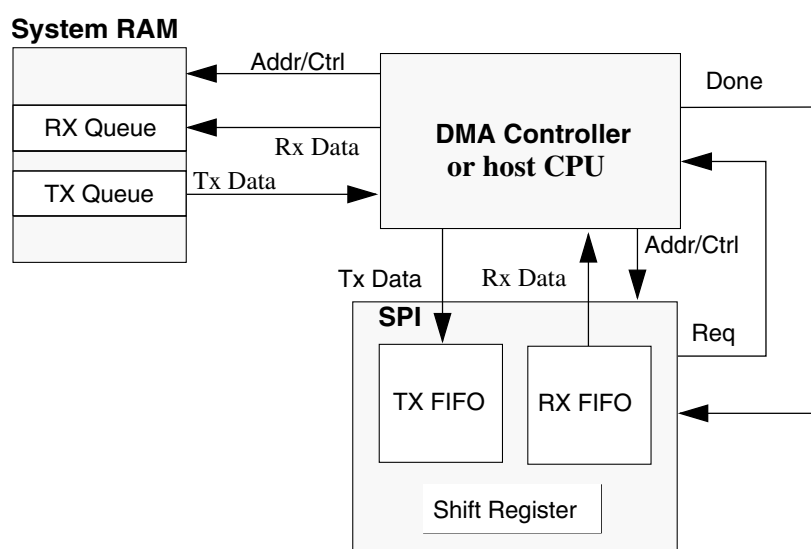


Figure 42-2. SPI with queues and DMA

42.2.3.2 DSI configuration

In the Deserial Serial Interface (DSI) configuration, DSPI serializes up to 32 parallel input signals or register bits. DSPI also deserializes the received data to parallel output signals or to a memory-mapped register. DSPI transfers the data using an SPI-like protocol.

Timed Serial Bus (TSB) mode provides the MicroSecond Channel (MSC) downstream support, serializing from 4 to 32 parallel input signals or register bits. For more information on TSB mode, see [Timed Serial Bus \(TSB\)](#).

42.2.3.3 CSI configuration

The Combined Serial Interface (CSI) configuration supports further configuration as follows:

- Normal: A combination of the SPI and DSI configurations. The DSPI interleaves DSI data frames with SPI data frames. Interleaving is done on the frame boundaries.
- TSB: Transmission of SPI data has higher priority than DSI data.

42.2.4 Modes of Operation

The module supports the following modes of operation that can be divided into two categories:

- Module-specific modes:
 - Master mode
 - Slave mode
 - Module Disable mode
- Chip-specific modes:
 - External Stop mode
 - Debug mode

The module enters module-specific modes when the host writes a module register. The chip-specific modes are controlled by signals external to the module. The chip-specific modes are modes that a chip may enter in parallel to the block-specific modes.

42.2.4.1 Master Mode

Master mode allows the module to initiate and control serial communication. In this mode, these signals are controlled by the module and configured as outputs:

- SCK
- PCS[x]

42.2.4.2 Slave Mode

Slave mode allows the module to communicate with SPI/DSI bus masters. In this mode, the module responds to externally controlled serial transfers. The SCK signal and the PCS[0]/ $\overline{SS}$ signals are configured as inputs and driven by an SPI bus master.

42.2.4.3 Module Disable Mode

The Module Disable mode can be used for chip power management. The clock to the non-memory mapped logic in the module can be stopped while in the Module Disable mode.

42.2.4.4 External Stop Mode

External Stop mode is used for chip power management. The module supports the Peripheral Bus Stop mode mechanism. When a request is made to enter External Stop mode, it acknowledges the request and completes the transfer that is in progress. When the module reaches the frame boundary, it signals that the protocol clock to the module may be shut off.

42.2.4.5 Debug Mode

Debug mode is used for system development and debugging. The MCR[FRZ] bit controls module behavior in the Debug mode:

- If the bit is set, the module stops all serial transfers, when the chip is in debug mode.
- If the bit is cleared, the chip debug mode has no effect on the module.

42.3 Module signal descriptions

This table describes the signals on the boundary of the module that may connect off chip (in alphabetical order).

Table 42-4. Module signal descriptions

Signal	Master mode	Slave mode	I/O
HT	Hardware Trigger	Hardware Trigger	I
PCS0/ $\overline{SS}$	Peripheral Chip Select 0 (O)	Slave Select (I)	I/O
PCS[1:3]	Peripheral Chip Selects 1–3	(Unused)	O
PCS4	Peripheral Chip Select 4	(Unused)	O

Table continues on the next page...

Table 42-4. Module signal descriptions (continued)

Signal	Master mode	Slave mode	I/O
PCS5/ $\overline{\text{PCSS}}$	Peripheral Chip Select 5 /Peripheral Chip Select Strobe	(Unused)	O
SCK	Serial Clock (O)	Serial Clock (I)	I/O
SIN	Serial Data In	Serial Data In	I
SOUT	Serial Data Out	Serial Data Out	O

42.3.1 HT—Hardware Trigger

Master mode: Hardware Trigger (I)—Honored only when the module is configured for the DSI or CSI configuration (MCR[DCONF]), hardware-trigger reception is enabled (DSICR0[TRRE] or TRIG[TRRE]), and the module is in the Running state (SR[TXRXS]). Receives hardware triggers that initiate data transfers. DSICR0[TPOL] specifies whether the module uses rising or falling edges of HT as hardware triggers.

Slave mode: Hardware Trigger (I)—Honored only when the module is configured for the DSI or CSI configuration (MCR[DCONF]), hardware-trigger reception is enabled (DSICR0[TRRE] or TRIG[TRRE]), and the module is in the Running state (SR[TXRXS]). DSICR0[TPOL] specifies whether the module uses rising or falling edges of HT as hardware triggers.

42.3.2 PCS0/ $\overline{\text{SS}}$ —Peripheral Chip Select/Slave Select

Master mode: Peripheral Chip Select 0 (O)—Selects an SPI slave to receive data transmitted from the module.

Slave mode: Slave Select (I)—Selects the module to receive data transmitted from an SPI master.

NOTE

Do not tie the DSPI slave select pin to ground. Otherwise, DSPI cannot function properly.

42.3.3 PCS1–PCS3—Peripheral Chip Selects 1–3

Master mode: Peripheral Chip Selects 1–3 (O)—Select an SPI slave to receive data transmitted by the module.

Slave mode: Unused

42.3.4 PCS4—Peripheral Chip Select 4

Master mode: Peripheral Chip Select 4 (O)—Selects an SPI slave to receive data transmitted by the module.

Slave mode: Unused

42.3.5 PCS5/ $\overline{\text{PCSS}}$ —Peripheral Chip Select 5/Peripheral Chip Select Strobe

Master mode:

- Peripheral Chip Select 5 (O)—Used only when the peripheral-chip-select strobe is disabled (MCR[PCSSE]). Selects an SPI slave to receive data transmitted by the module.
- Peripheral Chip Select Strobe (O)—Used only when the peripheral-chip-select strobe is enabled (MCR[PCSSE]). Strokes an off-module peripheral-chip-select demultiplexer, which decodes the module's PCS signals other than PCS5, preventing glitches on the demultiplexer outputs.

Slave mode: Unused

42.3.6 SCK—Serial Clock

Master mode: Serial Clock (O)—Supplies a clock signal from the module to SPI slaves.

Slave mode: Serial Clock (I)—Supplies a clock signal to the module from an SPI master.

42.3.7 SIN—Serial Input

Master mode: Serial Input (I)—Receives serial data.

Slave mode: Serial Input (I)—Receives serial data.

42.3.8 SOUT—Serial Output

Master mode: Serial Output (O)—Transmits serial data.

Slave mode: Serial Output (O)—Transmits serial data.

42.4 Memory Map/Register Definition

Register accesses to memory addresses that are reserved or undefined result in a transfer error. Any Write access to the POPR and RXFRn also results in a transfer error.

NOTE

While the module is in the running state, do not write to these registers:

- PISRn
- DIMRn, DPIRn, and SSRn

DSPI memory map

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/page
0	Module Configuration Register (DSPI_MCR)	32	R/W	0000_0001h	42.4.1/2152
8	Transfer Count Register (DSPI_TCR)	32	R/W	0000_0000h	42.4.2/2155
C	Clock and Transfer Attributes Register (In Master Mode) (DSPI_CTAR0)	32	R/W	7800_0000h	42.4.3/2156
C	Clock and Transfer Attributes Register (In Slave Mode) (DSPI_CTAR0_SLAVE)	32	R/W	7800_0000h	42.4.4/2161
10	Clock and Transfer Attributes Register (In Master Mode) (DSPI_CTAR1)	32	R/W	7800_0000h	42.4.3/2156
10	Clock and Transfer Attributes Register (In Slave Mode) (DSPI_CTAR1_SLAVE)	32	R/W	7800_0000h	42.4.4/2161
14	Clock and Transfer Attributes Register (In Master Mode) (DSPI_CTAR2)	32	R/W	7800_0000h	42.4.3/2156
18	Clock and Transfer Attributes Register (In Master Mode) (DSPI_CTAR3)	32	R/W	7800_0000h	42.4.3/2156
1C	Clock and Transfer Attributes Register (In Master Mode) (DSPI_CTAR4)	32	R/W	7800_0000h	42.4.3/2156
20	Clock and Transfer Attributes Register (In Master Mode) (DSPI_CTAR5)	32	R/W	7800_0000h	42.4.3/2156
24	Clock and Transfer Attributes Register (In Master Mode) (DSPI_CTAR6)	32	R/W	7800_0000h	42.4.3/2156
28	Clock and Transfer Attributes Register (In Master Mode) (DSPI_CTAR7)	32	R/W	7800_0000h	42.4.3/2156
2C	Status Register (DSPI_SR)	32	R/W	0201_0000h	42.4.5/2163
30	DMA/Interrupt Request Select and Enable Register (DSPI_RSER)	32	R/W	0000_0000h	42.4.6/2166
34	PUSH TX FIFO Register In Master Mode (DSPI_PUSHR)	32	R/W	0000_0000h	42.4.7/2168
34	PUSH TX FIFO Register In Slave Mode (DSPI_PUSHR_SLAVE)	32	R/W	0000_0000h	42.4.8/2170
38	POP RX FIFO Register (DSPI_POPR)	32	R	0000_0000h	42.4.9/2171

Table continues on the next page...

DSPI memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
3C	Transmit FIFO Registers (DSPI_TXFR0)	32	R	0000_0000h	42.4.10/ 2171
40	Transmit FIFO Registers (DSPI_TXFR1)	32	R	0000_0000h	42.4.10/ 2171
44	Transmit FIFO Registers (DSPI_TXFR2)	32	R	0000_0000h	42.4.10/ 2171
48	Transmit FIFO Registers (DSPI_TXFR3)	32	R	0000_0000h	42.4.10/ 2171
7C	Receive FIFO Registers (DSPI_RXFR0)	32	R	0000_0000h	42.4.11/ 2172
80	Receive FIFO Registers (DSPI_RXFR1)	32	R	0000_0000h	42.4.11/ 2172
84	Receive FIFO Registers (DSPI_RXFR2)	32	R	0000_0000h	42.4.11/ 2172
88	Receive FIFO Registers (DSPI_RXFR3)	32	R	0000_0000h	42.4.11/ 2172
BC	DSI Configuration Register 0 (DSPI_DSICR0)	32	R/W	0000_0000h	42.4.12/ 2173
C0	DSI Serialization Data Register 0 (DSPI_SDR0)	32	R	0000_0000h	42.4.13/ 2175
C4	DSI Alternate Serialization Data Register 0 (DSPI_ASDR0)	32	R/W	0000_0000h	42.4.14/ 2176
C8	DSI Transmit Comparison Register 0 (DSPI_COMPR0)	32	R	0000_0000h	42.4.15/ 2176
CC	DSI Deserialization Data Register 0 (DSPI_DDR0)	32	R	0000_0000h	42.4.16/ 2177
D0	DSI Configuration Register 1 (DSPI_DSICR1)	32	R/W	0000_0000h	42.4.17/ 2177
D4	DSI Serialization Source Select Register 0 (DSPI_SSR0)	32	R/W	0000_0000h	42.4.18/ 2179
D8	DSI Parallel Input Select Register 0 (DSPI_PISR0)	32	R/W	0000_0000h	42.4.19/ 2179
DC	DSI Parallel Input Select Register 1 (DSPI_PISR1)	32	R/W	0000_0000h	42.4.20/ 2180
E0	DSI Parallel Input Select Register 2 (DSPI_PISR2)	32	R/W	0000_0000h	42.4.21/ 2181
E4	DSI Parallel Input Select Register 3 (DSPI_PISR3)	32	R/W	0000_0000h	42.4.22/ 2182
E8	DSI Deserialized Data Interrupt Mask Register 0 (DSPI_DIMR0)	32	R/W	0000_0000h	42.4.23/ 2183
EC	DSI Deserialized Data Polarity Interrupt Register 0 (DSPI_DPIR0)	32	R/W	0000_0000h	42.4.24/ 2183

42.4.1 Module Configuration Register (DSPI_MCR)

Contains bits to configure various attributes associated with the module operations. The HALT and MDIS bits can be changed at any time, but the effect takes place only on the next frame boundary. Only the HALT and MDIS bits in the MCR can be changed, while the module is in the Running state.

Address: 0h base + 0h offset = 0h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R																
W	MSTR	CONT_SCKE		DCONF	FRZ	MTFE	PCSSE	ROOE	Reserved					PC SIS		
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R					0	0					0					
W	Reserved	MDIS	DIS_TXF	DIS_RXF	CLR_TXF	CLR_RXF	SMPL_PT							Reserved	PES	HALT
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1

DSPI_MCR field descriptions

Field	Description
0 MSTR	Master/Slave Mode Select Enables either Master mode (if supported) or Slave mode (if supported) operation. 0 Enables Slave mode 1 Enables Master mode
1 CONT_SCKE	Continuous SCK Enable Enables the Serial Communication Clock (SCK) to run continuously.

Table continues on the next page...

DSPI_MCR field descriptions (continued)

Field	Description
	0 Continuous SCK disabled. 1 Continuous SCK enabled.
2–3 DCONF	Interface Configuration Selects among the different interface configurations of the module. 00 SPI 01 DSI 10 CSI 11 Reserved
4 FRZ	Freeze Enables transfers to be stopped on the next frame boundary when the device enters Debug mode. 0 Do not halt serial transfers in Debug mode. 1 Halt serial transfers in Debug mode.
5 MTFE	Modified Transfer Format Enable Enables a modified transfer format to be used. 0 Modified SPI transfer format disabled. 1 Modified SPI transfer format enabled.
6 PCSSE	Peripheral Chip Select Strobe Enable Enables the PCS5/ $\overline{\text{PCSS}}$ to operate as a PCS Strobe output signal. 0 PCS5/ $\overline{\text{PCSS}}$ is used as the Peripheral Chip Select[5] signal. 1 PCS5/ $\overline{\text{PCSS}}$ is used as an active-low PCS Strobe signal.
7 ROOE	Receive FIFO Overflow Overwrite Enable In the RX FIFO overflow condition, configures the module to ignore the incoming serial data or overwrite existing data. If the RX FIFO is full and new data is received, the data from the transfer, generating the overflow, is ignored or shifted into the shift register. 0 Incoming data is ignored. 1 Incoming data is shifted into the shift register.
8–9 Reserved	Always write the reset value to this field. This field is reserved.
10–15 PCSIS	Peripheral Chip Select x Inactive State Determines the inactive state of PCSx. Refer to the chip-specific D SPI information for the number of PCS signals used in this chip. NOTE: The effect of this bit only takes place when module is enabled. Ensure that this bit is configured correctly before enabling the DSPI interface. 0 The inactive state of PCSx is low. 1 The inactive state of PCSx is high.
16 Reserved	This field is reserved.

Table continues on the next page...

DSPI_MCR field descriptions (continued)

Field	Description
17 MDIS	Module Disable Allows the clock to be stopped to the non-memory mapped logic in the module effectively putting it in a software-controlled power-saving state. The reset value of the MDIS bit is parameterized, with a default reset value of 0. When the module is used in Slave Mode, it is recommended to leave this bit 0, because a slave doesn't have control over master transactions. 0 Enables the module clocks. 1 Allows external logic to disable the module clocks.
18 DIS_TXF	Disable Transmit FIFO When the TX FIFO is disabled, the transmit part of the module operates as a simplified double-buffered SPI. This bit can be written only when the MDIS bit is cleared. 0 TX FIFO is enabled. 1 TX FIFO is disabled.
19 DIS_RXF	Disable Receive FIFO When the RX FIFO is disabled, the receive part of the module operates as a simplified double-buffered SPI. This bit can only be written when the MDIS bit is cleared. 0 RX FIFO is enabled. 1 RX FIFO is disabled.
20 CLR_TXF	Clear TX FIFO Flushes the TX FIFO. Writing a 1 to CLR_TXF clears the TX FIFO Counter. The CLR_TXF bit is always read as zero. 0 Do not clear the TX FIFO counter. 1 Clear the TX FIFO counter.
21 CLR_RXF	CLR_RXF Flushes the RX FIFO. Writing a 1 to CLR_RXF clears the RX Counter. The CLR_RXF bit is always read as zero. 0 Do not clear the RX FIFO counter. 1 Clear the RX FIFO counter.
22–23 SMPL_PT	Sample Point Controls when the module master samples SIN in Modified Transfer Format. This field is valid only when CPHA bit in CTARn[CPHA] is 0. 00 0 protocol clock cycles between SCK edge and SIN sample 01 1 protocol clock cycle between SCK edge and SIN sample 10 2 protocol clock cycles between SCK edge and SIN sample 11 Reserved
24–28 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
29 Reserved	This field is reserved.

Table continues on the next page...

DSPI_MCR field descriptions (continued)

Field	Description
30 PES	Parity Error Stop Controls SPI operation when a parity error is detected in a received SPI frame. 0 SPI frame transmission continues. 1 SPI frame transmission stops.
31 HALT	Halt The HALT bit starts and stops frame transfers. See Start and Stop of Module transfers 0 Start transfers. 1 Stop transfers.

42.4.2 Transfer Count Register (DSPI_TCR)

TCR contains a counter that indicates the number of SPI transfers made. The transfer counter is intended to assist in queue management. Do not write the TCR when the module is in the Running state.

Address: 0h base + 8h offset = 8h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R																																
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

DSPI_TCR field descriptions

Field	Description
0–15 SPI_TCNT	SPI Transfer Counter Counts the number of SPI transfers the module makes. The SPI_TCNT field increments every time the last bit of an SPI frame is transmitted. A value written to SPI_TCNT presets the counter to that value. SPI_TCNT is reset to zero at the beginning of the frame when the CTCNT field is set in the executing SPI command. The Transfer Counter wraps around; incrementing the counter past 65535 resets the counter to zero.
16–31 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

42.4.3 Clock and Transfer Attributes Register (In Master Mode) (DSPI_CTAR_n)

CTAR registers are used to define different transfer attributes. Do not write to the CTAR registers while the module is in the Running state.

In Master mode, the CTAR registers define combinations of transfer attributes such as frame size, clock phase and polarity, data bit ordering, baud rate, and various delays. In slave mode, a subset of the bitfields in CTAR0 and CTAR1 are used to set the slave transfer attributes.

When the module is configured as a SPI master, the CTAS field in the command portion of the TX FIFO entry selects which of the CTAR registers is used. When the module is configured as an SPI bus slave, it uses the CTAR0 register.

When the module is configured as a DSI master, the DSICTAS field in the DSI Configuration Register 0 (DSICR0), selects which of the CTAR register is used. When the module is configured as a DSI bus slave, the CTAR1 register is used.

In CSI Configuration, the transfer attributes are selected based on whether the current frame is SPI data or DSI data. SPI transfers in CSI Configuration follow the protocol described for SPI Configuration, and DSI transfers in CSI Configuration follow the protocol described for DSI Configuration. CSI Configuration is valid only in conjunction with master mode.

TSB mode sets some limitations on transfer attributes:

- Clock phase is forced to be CPHA = 1 and the CPHA bit setting has no effect.
- PCS lines are driven at the driving edge of the SCK clock together with SOUT, so PCS assertion and negation delays control is unavailable and PCSSCK, PASC, CSSCK and ASC fields have no effect.
- Delay after transfer can be set from 1 to 64 serial clocks with help of PDT and DT fields

Address: 0h base + Ch offset + (4d × i), where i=0d to 7d

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	DBR		FMSZ				CPOL	CPHA	LSBFE	PCSSCK		PASC		PDT		PBR
W																
Reset	0	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	CSSCK				ASC				DT				BR			
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

DSPI_CTAR_n field descriptions

Field	Description																																								
0 DBR	Double Baud Rate Doubles the effective baud rate of the Serial Communications Clock (SCK). This field is used only in master mode. It effectively halves the Baud Rate division ratio, supporting faster frequencies, and odd division ratios for the Serial Communications Clock (SCK). When the DBR bit is set, the duty cycle of the Serial Communications Clock (SCK) depends on the value in the Baud Rate Prescaler and the Clock Phase bit as listed in the following table. See the BR field description for details on how to compute the baud rate. Table 42-11. DSPI SCK Duty Cycle <table><tr><th>DBR</th><th>CPHA</th><th>PBR</th><th>SCK Duty Cycle</th></tr><tr><td>0</td><td>any</td><td>any</td><td>50/50</td></tr><tr><td>1</td><td>0</td><td>00</td><td>50/50</td></tr><tr><td>1</td><td>0</td><td>01</td><td>33/66</td></tr><tr><td>1</td><td>0</td><td>10</td><td>40/60</td></tr><tr><td>1</td><td>0</td><td>11</td><td>43/57</td></tr><tr><td>1</td><td>1</td><td>00</td><td>50/50</td></tr><tr><td>1</td><td>1</td><td>01</td><td>66/33</td></tr><tr><td>1</td><td>1</td><td>10</td><td>60/40</td></tr><tr><td>1</td><td>1</td><td>11</td><td>57/43</td></tr></table> 0 The baud rate is computed normally with a 50/50 duty cycle. 1 The baud rate is doubled with the duty cycle depending on the Baud Rate Prescaler.	DBR	CPHA	PBR	SCK Duty Cycle	0	any	any	50/50	1	0	00	50/50	1	0	01	33/66	1	0	10	40/60	1	0	11	43/57	1	1	00	50/50	1	1	01	66/33	1	1	10	60/40	1	1	11	57/43
DBR	CPHA	PBR	SCK Duty Cycle																																						
0	any	any	50/50																																						
1	0	00	50/50																																						
1	0	01	33/66																																						
1	0	10	40/60																																						
1	0	11	43/57																																						
1	1	00	50/50																																						
1	1	01	66/33																																						
1	1	10	60/40																																						
1	1	11	57/43																																						
1–4 FMSZ	Frame Size The number of bits transferred per frame is equal to the FMSZ value plus 1. Regardless of the transmission mode, the minimum valid frame size value is 4. There is a constraint on the minimum allowable Frame size, which depends on the ratio of the Register Read/Write clock to the Protocol clock. The minimum Frame size (FMSZ + 1) is determined by the following equation. Upper Ceiling must be applied for non-integer values. Minimum Frame Size = ((4 x f_p) + (3 x f_r)) / (n x f_r) f_p = Protocol clock frequency f_r = Register interface clock frequency n = (PBR x BR) / (1 + DBR) = Multiple of protocol clock required to get Baud (SCK) Clock However, the minimum frame size can never be less than 4. There is no constraint on the maximum programmable frame size. Also note that 'fp' can be equal to, less than or greater than 'fr' purely based on user requirement.																																								

Table continues on the next page...

DSPI_CTAR_n field descriptions (continued)

Field	Description
5 CPOL	Clock Polarity Selects the inactive state of the Serial Communications Clock (SCK). This bit is used in both master and slave mode. For successful communication between serial devices, the devices must have identical clock polarities. When the Continuous Selection Format is selected, switching between clock polarities without stopping the module can cause errors in the transfer due to the peripheral device interpreting the switch of clock polarity as a valid clock edge. NOTE: In case of Continuous SCK mode, when the module goes in low power mode(disabled), inactive state of SCK is not guaranteed. 0 The inactive state value of SCK is low. 1 The inactive state value of SCK is high.
6 CPHA	Clock Phase Selects which edge of SCK causes data to change and which edge causes data to be captured. This bit is used in both master and slave mode. For successful communication between serial devices, the devices must have identical clock phase settings. In Continuous SCK mode or TSB mode, the bit value is ignored and the transfers are done as if the CPHA bit is set to 1. 0 Data is captured on the leading edge of SCK and changed on the following edge. 1 Data is changed on the leading edge of SCK and captured on the following edge.
7 LSBFE	LSB First Specifies whether the LSB or MSB of the frame is transferred first. 0 Data is transferred MSB first. 1 Data is transferred LSB first.
8–9 PCSSCK	PCS to SCK Delay Prescaler Selects the prescaler value for the delay between assertion of PCS and the first edge of the SCK. See the CSSCK field description for information on how to compute the PCS to SCK Delay. Refer PCS to SCK Delay (t_{CSC}) for more details. 00 PCS to SCK Prescaler value is 1. 01 PCS to SCK Prescaler value is 3. 10 PCS to SCK Prescaler value is 5. 11 PCS to SCK Prescaler value is 7.
10–11 PASC	After SCK Delay Prescaler Selects the prescaler value for the delay between the last edge of SCK and the negation of PCS. See the ASC field description for information on how to compute the After SCK Delay. In the TSB mode, the PASC field has no effect. Refer After SCK Delay (t_{ASC}) for more details. 00 Delay after Transfer Prescaler value is 1. 01 Delay after Transfer Prescaler value is 3. 10 Delay after Transfer Prescaler value is 5. 11 Delay after Transfer Prescaler value is 7.
12–13 PDT	Delay after Transfer Prescaler Selects the prescaler value for the delay between the negation of the PCS signal at the end of a frame and the assertion of PCS at the beginning of the next frame. The PDT field is only used in master mode. See

Table continues on the next page...

DSPI_CTAR_n field descriptions (continued)

Field	Description																																		
	the DT field description for details on how to compute the Delay after Transfer. Refer Delay after Transfer (t_{DT}) for more details. 00 Delay after Transfer Prescaler value is 1. 01 Delay after Transfer Prescaler value is 3. 10 Delay after Transfer Prescaler value is 5. 11 Delay after Transfer Prescaler value is 7.																																		
14–15 PBR	Baud Rate Prescaler Selects the prescaler value for the baud rate. This field is used only in master mode. The baud rate is the frequency of the SCK. The protocol clock is divided by the prescaler value before the baud rate selection takes place. See the BR field description for details on how to compute the baud rate. 00 Baud Rate Prescaler value is 2. 01 Baud Rate Prescaler value is 3. 10 Baud Rate Prescaler value is 5. 11 Baud Rate Prescaler value is 7.																																		
16–19 CSSCK	PCS to SCK Delay Scaler Selects the scaler value for the PCS to SCK delay. This field is used only in master mode. In the TSB mode the field has no effect. The PCS to SCK Delay is the delay between the assertion of PCS and the first edge of the SCK. The delay is a multiple of the protocol clock period, and it is computed according to the following equation: $t_{CSC} = (1/f_P) \times PCSSCK \times CSSCK.$ The following table lists the delay scaler values. Table 42-10. Delay Scaler Encoding <table> <tr> <th>Field Value</th><th>Delay Scaler Value</th></tr> <tr><td>0000</td><td>2</td></tr> <tr><td>0001</td><td>4</td></tr> <tr><td>0010</td><td>8</td></tr> <tr><td>0011</td><td>16</td></tr> <tr><td>0100</td><td>32</td></tr> <tr><td>0101</td><td>64</td></tr> <tr><td>0110</td><td>128</td></tr> <tr><td>0111</td><td>256</td></tr> <tr><td>1000</td><td>512</td></tr> <tr><td>1001</td><td>1024</td></tr> <tr><td>1010</td><td>2048</td></tr> <tr><td>1011</td><td>4096</td></tr> <tr><td>1100</td><td>8192</td></tr> <tr><td>1101</td><td>16384</td></tr> <tr><td>1110</td><td>32768</td></tr> <tr><td>1111</td><td>65536</td></tr> </table>	Field Value	Delay Scaler Value	0000	2	0001	4	0010	8	0011	16	0100	32	0101	64	0110	128	0111	256	1000	512	1001	1024	1010	2048	1011	4096	1100	8192	1101	16384	1110	32768	1111	65536
Field Value	Delay Scaler Value																																		
0000	2																																		
0001	4																																		
0010	8																																		
0011	16																																		
0100	32																																		
0101	64																																		
0110	128																																		
0111	256																																		
1000	512																																		
1001	1024																																		
1010	2048																																		
1011	4096																																		
1100	8192																																		
1101	16384																																		
1110	32768																																		
1111	65536																																		

Table continues on the next page...

DSPI_CTARn field descriptions (continued)

Field	Description																												
	Refer PCS to SCK Delay (t_{CSC}) for more details.																												
20–23 ASC	After SCK Delay Scaler Selects the scaler value for the After SCK Delay. This field is used only in master mode. The After SCK Delay is the delay between the last edge of SCK and the negation of PCS. The delay is a multiple of the protocol clock period, and it is computed according to the following equation: $t_{ASC} = (1/f_P) \times PASC \times ASC$ See Delay Scaler Encoding table in CTARn[CSSCK] bit field description for scaler values. Refer After SCK Delay (t_{ASC}) for more details.																												
24–27 DT	Delay After Transfer Scaler Selects the Delay after Transfer Scaler. This field is used only in master mode. The Delay after Transfer is the time between the negation of the PCS signal at the end of a frame and the assertion of PCS at the beginning of the next frame. In the Continuous Serial Communications Clock operation, the DT value is fixed to one SCK clock period, The Delay after Transfer is a multiple of the protocol clock period, and it is computed according to the following equation: $t_{DT} = (1/f_P) \times PDT \times DT$ See Delay Scaler Encoding table in CTARn[CSSCK] bit field description for scaler values. In the TSB mode, the Delay after Transfer is equal to a number formed by concatenation of the PDT and DT fields plus 1 of the SCK clock periods. Refer Delay after Transfer (t_{DT}) for more details.																												
28–31 BR	Baud Rate Scaler Selects the scaler value for the baud rate. This field is used only in master mode. The prescaled protocol clock is divided by the Baud Rate Scaler to generate the frequency of the SCK. The baud rate is computed according to the following equation: $SCK \text{ baud rate} = (f_P / PBR) \times [(1+DBR)/BR]$ The following table lists the baud rate scaler values. Table 42-9. Baud Rate Scaler <table> <tr> <th>CTARn[BR]</th><th>Baud Rate Scaler Value</th></tr> <tr><td>0000</td><td>2</td></tr> <tr><td>0001</td><td>4</td></tr> <tr><td>0010</td><td>6</td></tr> <tr><td>0011</td><td>8</td></tr> <tr><td>0100</td><td>16</td></tr> <tr><td>0101</td><td>32</td></tr> <tr><td>0110</td><td>64</td></tr> <tr><td>0111</td><td>128</td></tr> <tr><td>1000</td><td>256</td></tr> <tr><td>1001</td><td>512</td></tr> <tr><td>1010</td><td>1024</td></tr> <tr><td>1011</td><td>2048</td></tr> <tr><td>1100</td><td>4096</td></tr> </table>	CTARn[BR]	Baud Rate Scaler Value	0000	2	0001	4	0010	6	0011	8	0100	16	0101	32	0110	64	0111	128	1000	256	1001	512	1010	1024	1011	2048	1100	4096
CTARn[BR]	Baud Rate Scaler Value																												
0000	2																												
0001	4																												
0010	6																												
0011	8																												
0100	16																												
0101	32																												
0110	64																												
0111	128																												
1000	256																												
1001	512																												
1010	1024																												
1011	2048																												
1100	4096																												

Table continues on the next page...

DSPI_CTAR_n field descriptions (continued)

Field	Description
Table 42-9. Baud Rate Scaler (continued)	
CTAR _n [BR]	Baud Rate Scaler Value
1101	8192
1110	16384
1111	32768

42.4.4 Clock and Transfer Attributes Register (In Slave Mode) (DSPI_CTAR_n_SLAVE)

When the module is configured as an SPI bus slave, the CTAR0 register is used.

When the module is configured as a DSI master, the DSICTAS field in the DSI Configuration Register 0 (DSICR0), selects which of the CTAR register is used. When the module is configured as a DSI bus slave, the CTAR1 register is used.

Address: 0h base + Ch offset + (4d × i), where i=0d to 1d

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	FMSZ					CPOL	CPHA	PE	PP	Reserved	Reserved					
W																
Reset	0	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0
Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	Reserved															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

DSPI_CTAR_n_SLAVE field descriptions

Field	Description
0–4 FMSZ	Frame Size The number of bits transferred per frame is equal to the FMSZ field value plus 1. Note that the minimum valid value of frame size is 4.
5 CPOL	Clock Polarity Selects the inactive state of the Serial Communications Clock (SCK).

Table continues on the next page...

DSPI_CTARn_SLAVE field descriptions (continued)

Field	Description
	NOTE: In case of Continuous SCK mode, when the module goes in low power mode(disabled), inactive state of SCK is not guaranteed. 0 The inactive state value of SCK is low. 1 The inactive state value of SCK is high.
6 CPHA	Clock Phase Selects which edge of SCK causes data to change and which edge causes data to be captured. This bit is used in both master and slave mode. For successful communication between serial devices, the devices must have identical clock phase settings. In Continuous SCK mode or TSB mode, the bit value is ignored and the transfers are done as if the CPHA bit is set to 1. 0 Data is captured on the leading edge of SCK and changed on the following edge. 1 Data is changed on the leading edge of SCK and captured on the following edge.
7 PE	Parity Enable Enables parity bit transmission and reception for the frame. 0 No parity bit included/checked. 1 Parity bit is transmitted instead of last data bit in frame, parity checked for received frame.
8 PP	Parity Polarity Controls polarity of the parity bit transmitted and checked. 0 Even Parity: the number of 1 bits in the transmitted frame is even. The SR[SPEF] bit is set if the number of 1 bits is odd in the received frame. 1 Odd Parity: the number of 1 bits in the transmitted frame is odd. The SR[SPEF] bit is set if the number of 1 bits is even in the received frame.
9 Reserved	This field is reserved.
10–31 Reserved	This field is reserved.

42.4.5 Status Register (DSPI_SR)

SR contains status and flag bits. The bits reflect the status of the module and indicate the occurrence of events that can generate interrupt or DMA requests. Software can clear flag bits in the SR by writing a 1 to them. Writing a 0 to a flag bit has no effect. This register may not be writable in Module Disable mode due to the use of power saving mechanisms.

Address: 0h base + 2Ch offset = 2Ch

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	TCF	TXRXS	Reserved	EOQF	TFUF	Reserved	TFFF	0	0	DPEF	SPEF	DDIF	RFOF	0	RFDF	Reserved
W	w1c		w1c	w1c	w1c	w1c	w1c			w1c	w1c	w1c	w1c		w1c	w1c
Reset	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	1

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	TXCTR				TXNXTPTR				RXCTR				POPNTPTTR			
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

DSPI_SR field descriptions

Field	Description
0 TCF	Transfer Complete Flag Indicates that all bits in a frame have been shifted out. TCF remains set until it is cleared by writing a 1 to it. 0 Transfer not complete. 1 Transfer complete.

Table continues on the next page...

DSPI_SR field descriptions (continued)

Field	Description
1 TXRXS	TX and RX Status Reflects the run status of the module. 0 Transmit and receive operations are disabled (The module is in Stopped state). 1 Transmit and receive operations are enabled (The module is in Running state).
2 Reserved	This field is reserved.
3 EOQF	End of Queue Flag Indicates that the last entry in a queue has been transmitted when the module is in Master mode. The EOQF bit is set when the TX FIFO entry has the EOQ bit set in the command halfword and the end of the transfer is reached. The EOQF bit remains set until cleared by writing a 1 to it. When the EOQF bit is set, the TXRXS bit is automatically cleared. 0 EOQ is not set in the executing command. 1 EOQ is set in the executing SPI command.
4 TFUF	Transmit FIFO Underflow Flag Indicates an underflow condition in the TX FIFO. The transmit underflow condition is detected only for DSPI blocks operating in Slave mode and SPI configuration. TFUF is set when the TX FIFO of the module operating in SPI Slave mode is empty and an external SPI master initiates a transfer. The TFUF bit remains set until cleared by writing 1 to it. 0 No TX FIFO underflow. 1 TX FIFO underflow has occurred.
5 Reserved	This field is reserved.
6 TFFF	Transmit FIFO Fill Flag Indicates whether there is an available location to be filled in the FIFO. Either a DMA request or an interrupt indication can be used to add another entry to the FIFO. Note that this bit is set if at least one location is free in the FIFO. The TFFF bit can be cleared by writing 1 to it or by acknowledgement from the DMA controller to the TX FIFO full request. NOTE: The reset value of this bit is 0 when the module is disabled, (MCR[MDIS]=1). 0 TX FIFO is full. 1 TX FIFO is not full.
7 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
8 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
9 DPEF	DSI Parity Error Flag Indicates that a DSI frame with parity error had been received. The bit remains set until it is cleared by writing a 1 to it. 0 No parity error. 1 Parity error has occurred.
10 SPEF	SPI Parity Error Flag

Table continues on the next page...

DSPI_SR field descriptions (continued)

Field	Description
	Indicates that a SPI frame with parity error had been received. The bit remains set until it is cleared by writing a 1 to it. 0 No parity error. 1 Parity error has occurred.
11 DDIF	DSI Data Received with Active Bits Indicates that a DSI frame had been received with bits selected by DIMR with active polarity, defined by the DPIR register. The bit remains set until it is cleared by writing a 1 to it. 0 No DSI data with active bits was received. 1 DSI data with active bits was received.
12 RFOF	Receive FIFO Overflow Flag Indicates an overflow condition in the RX FIFO. The field is set when the RX FIFO and shift register are full and a transfer is initiated. The bit remains set until it is cleared by writing a 1 to it. 0 No Rx FIFO overflow. 1 Rx FIFO overflow has occurred.
13 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
14 RFDF	Receive FIFO Drain Flag Indicates whether there is an available location to be drained from the FIFO. Either a DMA request or an interrupt indication can be used to read from the FIFO. Note that this bit is set if at least one location can be read from the FIFO. The RFDF bit can be cleared by writing 1 to it or by acknowledgement from the DMA controller when the RX FIFO is empty. 0 RX FIFO is empty. 1 RX FIFO is not empty.
15 Reserved	This field is reserved.
16–19 TXCTR	TX FIFO Counter Indicates the number of valid entries in the TX FIFO. The TXCTR is incremented every time the PUSHHR is written. The TXCTR is decremented every time an SPI command is executed and the SPI data is transferred to the shift register.
20–23 TXNXTPTR	Transmit Next Pointer Indicates which TX FIFO entry is transmitted during the next transfer. The TXNXTPTR field is updated every time SPI data is transferred from the TX FIFO to the shift register.
24–27 RXCTR	RX FIFO Counter Indicates the number of entries in the RX FIFO. The RXCTR is decremented every time the POPR is read. The RXCTR is incremented every time data is transferred from the shift register to the RX FIFO.
28–31 POPNXTPTR	Pop Next Pointer Contains a pointer to the RX FIFO entry to be returned when the POPR is read. The POPNXTPTR is updated when the POPR is read.

42.4.6 DMA/Interrupt Request Select and Enable Register (DSPI_RSER)

RSER controls DMA and interrupt requests. Do not write to the RSER while the module is in the Running state.

Address: 0h base + 30h offset = 30h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	TCF_RE	Reserved	Reserved	EOQF_RE	TFUF_RE	Reserved	TFFF_RE	TFFF_DIRS	Reserved	DPEF_RE	SPEF_RE	DDIF_RE	RFOF_RE	Reserved	RFDF_RE	RFDF_DIRS
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	Reserved	Reserved	0													
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

DSPI_RSER field descriptions

Field	Description
0 TCF_RE	Transmission Complete Request Enable Enables TCF flag in the SR to generate an interrupt request. 0 TCF interrupt requests are disabled. 1 TCF interrupt requests are enabled.
1 Reserved	Always write the reset value to this field. This field is reserved.
2 Reserved	This field is reserved.
3 EOQF_RE	Finished Request Enable Enables the EOQF flag in the SR to generate an interrupt request. 0 EOQF interrupt requests are disabled. 1 EOQF interrupt requests are enabled.
4 TFUF_RE	Transmit FIFO Underflow Request Enable Enables the TFUF flag in the SR to generate an interrupt request.

Table continues on the next page...

DSPI_RSER field descriptions (continued)

Field	Description
	0 TFUF interrupt requests are disabled. 1 TFUF interrupt requests are enabled.
5 Reserved	This field is reserved.
6 TFFF_RE	Transmit FIFO Fill Request Enable Enables the TFFF flag in the SR to generate a request. The TFFF_DIRS bit selects between generating an interrupt request or a DMA request. 0 TFFF interrupts or DMA requests are disabled. 1 TFFF interrupts or DMA requests are enabled.
7 TFFF_DIRS	Transmit FIFO Fill DMA or Interrupt Request Select Selects between generating a DMA request or an interrupt request. When SR[TFFF] and RSER[TFFF_RE] are set, this field selects between generating an interrupt request or a DMA request. 0 TFFF flag generates interrupt requests. 1 TFFF flag generates DMA requests.
8 Reserved	Always write the reset value to this field. This field is reserved.
9 DPEF_RE	DSI Parity Error Request Enable Enables the DPEF flag in the SR to generate an interrupt request. 0 DPEF interrupt requests are disabled. 1 DPEF interrupt requests are enabled.
10 SPEF_RE	SPI Parity Error Request Enable Enables the SPEF flag in the SR to generate an interrupt request. 0 SPEF interrupt requests are disabled. 1 SPEF interrupt requests are enabled.
11 DDIF_RE	DSI data received with active bits Request Enable Enables the SR[DDIF] flag to generate an interrupt request. 0 Disabled 1 Enabled
12 RFOF_RE	Receive FIFO Overflow Request Enable Enables the RFOF flag in the SR to generate an interrupt request. 0 RFOF interrupt requests are disabled. 1 RFOF interrupt requests are enabled.
13 Reserved	Always write the reset value to this field. This field is reserved.
14 RFDF_RE	Receive FIFO Drain Request Enable

Table continues on the next page...

DSPI_RSER field descriptions (continued)

Field	Description
	Enables the RFDF flag in the SR to generate a request. The RFDF_DIRS bit selects between generating an interrupt request or a DMA request. 0 RFDF interrupt or DMA requests are disabled. 1 RFDF interrupt or DMA requests are enabled.
15 RFDF_DIRS	Receive FIFO Drain DMA or Interrupt Request Select Selects between generating a DMA request or an interrupt request. When the RFDF flag bit in the SR is set, and the RFDF_RE bit in the RSER is set, the RFDF_DIRS bit selects between generating an interrupt request or a DMA request. 0 Interrupt request. 1 DMA request.
16 Reserved	Always write the reset value to this field. This field is reserved.
17 Reserved	Always write the reset value to this field. This field is reserved.
18–31 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

42.4.7 PUSH TX FIFO Register In Master Mode (DSPI_PUSHR)

Specifies data to be transferred to the TX FIFO and CMD FIFO. User must write 16-bits data into TXDATA field. An 8- or 16-bit write access to the TXDATA field transfers 16 bits of data bus to the TX FIFO. A write access to the command fields transfers the 16 bits of command information to the CMD FIFO. In Master mode, the register transfers 16 bits of data to the TX FIFO and 16 bits of command information to the CMD FIFO.

The TX FIFO and CMD FIFO must be filled simultaneously. In other words, you must perform write accesses to both the data and command fields for every PUSHR operation. Because both the TX FIFO and CMD FIFO are written to and read from simultaneously, they behave as a single 32 bit FIFO.

A read access of PUSHR returns the topmost TX FIFO and CMD FIFO entries concatenated.

When the module is disabled, writing to this register does not update the FIFO. Therefore, any reads performed while the module is disabled return the last PUSHR write performed while the module was still enabled.

Address: 0h base + 34h offset = 34h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	CONT	CTAS			EOQ	CTCNT	PE	PP	Reserved		PCS					
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	TXDATA															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

DSPI_PUSHHR field descriptions

Field	Description
0 CONT	Continuous Peripheral Chip Select Enable Selects a continuous selection format. The bit is used in SPI Master mode. The bit enables the selected PCS signals to remain asserted between transfers. 0 Return PCSn signals to their inactive state between transfers. 1 Keep PCSn signals asserted between transfers.
1–3 CTAS	Clock and Transfer Attributes Select Selects which CTAR to use in master mode to specify the transfer attributes for the associated SPI frame. In SPI Slave mode, CTAR0 is used. See the chip specific section for details to determine how many CTARs this device has. You should not program a value in this field for a register that is not present. 000 CTAR0 001 CTAR1 010 CTAR2 011 CTAR3 100 CTAR4 101 CTAR5 110 CTAR6 111 CTAR7
4 EOQ	End Of Queue Host software uses this bit to signal to the module that the current SPI transfer is the last in a queue. At the end of the transfer, the EOQF bit in the SR is set. 0 The SPI data is not the last data to transfer. 1 The SPI data is the last data to transfer.
5 CTCNT	Clear Transfer Counter Clears the TCNT field in the TCR register. The TCNT field is cleared before the module starts transmitting the current SPI frame. 0 Do not clear the TCR[TCNT] field. 1 Clear the TCR[TCNT] field.

Table continues on the next page...

DSPI_PUSHR field descriptions (continued)

Field	Description
6 PE	Parity Enable Enables parity bit transmission and parity reception check for the SPI frame. 0 No parity bit included/checked. 1 Parity bit is transmitted instead of the last data bit in the frame; parity is checked for the received frame.
7 PP	Parity Polarity Controls the polarity of the parity bit transmitted and checked. 0 Even Parity: the number of 1 bits in the transmitted frame is even. The SR[SPEF] bit is set if the number of 1 bits is odd in the received frame. 1 Odd Parity: the number of 1 bits in the transmitted frame is odd. The SR[SPEF] bit is set if the number of 1 bits is even in the received frame.
8–9 Reserved	Always write the reset value to this field. This field is reserved.
10–15 PCS	Select which PCS signals are to be asserted for the transfer. Refer to the chip-specific DSPI information for the number of PCS signals used in this chip. 0 Negate the PCS[x] signal. 1 Assert the PCS[x] signal.
16–31 TXDATA	Transmit Data Holds SPI data to be transferred according to the associated SPI command.

42.4.8 PUSH TX FIFO Register In Slave Mode (DSPI_PUSHR_SLAVE)

Specifies data to be transferred to the TX FIFO in slave mode. An 8- or 16-bit write access to PUSHR transfers the 16-bit TXDATA field to the TX FIFO.

Address: 0h base + 34h offset = 34h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0																TXDATA															
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	

DSPI_PUSHR_SLAVE field descriptions

Field	Description
0–15 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
16–31 TXDATA	Transmit Data Holds SPI data to be transferred according to the associated SPI command.

42.4.9 POP RX FIFO Register (DSPI_POPR)

POPR is used to read the RX FIFO. Eight- or sixteen-bit read accesses to the POPR have the same effect on the RX FIFO as 32-bit read accesses. A write to this register will generate a Transfer Error.

Address: 0h base + 38h offset = 38h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	RXDATA																															
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	

DSPI_POPR field descriptions

Field	Description
0–31 RXDATA	Received Data Contains the SPI data from the RX FIFO entry to which the Pop Next Data Pointer points.

42.4.10 Transmit FIFO Registers (DSPI_TXFRn)

TXFRn registers provide visibility into the TX FIFO for debugging purposes. Each register is an entry in the TX FIFO. The registers are read-only and cannot be modified. Reading the TXFRx registers does not alter the state of the TX FIFO.

Address: 0h base + 3Ch offset + (4d × i), where i=0d to 3d

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	TXCMD_TXDATA															TXDATA																
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	

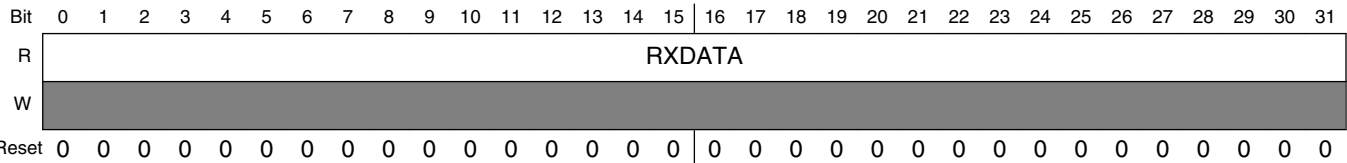
DSPI_TXFRn field descriptions

Field	Description
0–15 TXCMD_ TXDATA	Transmit Command or Transmit Data In Master mode the TXCMD field contains the command that sets the transfer attributes for the SPI data. In Slave mode, this field is reserved.
16–31 TXDATA	Transmit Data Contains the SPI data to be shifted out.

42.4.11 Receive FIFO Registers (DSPI_RXFRn)

RXFRn provide visibility into the RX FIFO for debugging purposes. Each register is an entry in the RX FIFO. The RXFR registers are read-only. Reading the RXFRx registers does not alter the state of the RX FIFO. The field MCR[MDIS] must be 0 when RXFR is read.

Address: 0h base + 7Ch offset + (4d × i), where i=0d to 3d



DSPI_RXFRn field descriptions

Field	Description
0–31 RXDATA	Receive Data Contains the received SPI data.

42.4.12 DSI Configuration Register 0 (DSPI_DSICR0)

Selects various attributes associated with the following configurations:

- DSI
- TSB

Do not write to the DSICR0 while the module is in the Running state.

Address: 0h base + BCh offset = BCh

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	Reserved	FMSZ4	Reserved						0	0						
W												TSBC	TXSS	TPOL	TRRE	CID
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	DCONT	DSICTAS				DMS	PES	PE	PP	DPCSx						
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

DSPI_DSICR0 field descriptions

Field	Description
0 Reserved	This field is reserved.
1 FMSZ4	MSB of the frame size in master mode. If the bit is set, 16 is added to the frame size, as defined by the CTARn[FMSZ] field. The CTARx register is selected by the DSICTAS field.
2–7 Reserved	This field is reserved.
8 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

Table continues on the next page...

DSPI_DSICR0 field descriptions (continued)

Field	Description
9–10 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
11 TSBC	Timed Serial Bus Configuration. Enables the Timed Serial Bus interface configuration and allows t_{DT} to be programmable. 0 Timed Serial Bus Configuration disabled. 1 Timed Serial Bus Configuration enabled.
12 TXSS	Transmit Data Source Select. Selects the source of data to be serialized. The source can be either data from host software written to the DSI Alternate Serialization Data Register (ASDR0/1) or Parallel Input pin states latched into the DSI Serialization Data Register (SDR0/1). 0 Source of serialized data is the SDR. 1 Source of serialized data is the ASDR.
13 TPOL	Trigger Polarity Selects the active edge of the hardware trigger input signal (HT), initiating DSI frames transfer. 0 Falling edge initiates a transfer. 1 Rising edge initiates a transfer.
14 TRRE	Trigger Reception Enable Enables the module to initiate a DSI frames transfer with an external trigger signal. NOTE: When TRIG[ENABLE] is set, the TRRE bit is given by TRIG[TRRE] which overrides DSICR0[TRRE]. 0 Trigger signal reception disabled. 1 Trigger signal reception enabled.
15 CID	Change In Data Transfer Enable Enables a change in serialization data to initiate a DSI frames transfer in DSI and CSI configurations. When the CID bit is set, DSI frames are initiated when the current DSI data differs from the previous DSI data shifted out. NOTE: When TRIG[ENABLE] is set, the CID bit is given by TRIG[CID] which overrides DSICR0[CID].
16 DCONT	DSI Continuous Peripheral Chip Select Enable Enables the PCS signals to remain asserted between transfers. The DCONT bit affects the PCS signals only in DSI master mode. When the TSBC bit is set, DCONT has no effect. 0 Return Peripheral Chip Select signals to their inactive state after the transfer is complete. 1 Keep Peripheral Chip Select signals asserted after transfer is complete.
17–19 DSICTAS	DSI Clock and Transfer Attributes Select Selects the CTAR register to use to provide transfer attributes for DSI frames. The DSICTAS field is used in DSI master mode. In DSI slave mode, CTAR1 is always selected.
20 DMS	Data Match Stop If set, stops DSI frame transmissions if DDIF flag is set in the SR register.

Table continues on the next page...

DSPI_DSICR0 field descriptions (continued)

Field	Description
	0 DDIF flag does not have an effect on DSI frame transmissions. 1 DDIF flag stops DSI frame transmissions.
21 PES	Parity Error Stop If set, stops DSI operation if the parity error happened in the received DSI frame. 0 Parity error does not stop DSI frame transmissions. 1 Parity error stops all DSI frame transmissions.
22 PE	Parity Enable Enables parity bit transmission and parity reception check for the DSI frames. 0 No parity bit included/checked. 1 Parity bit is transmitted instead of the last data bit in the frame; parity is checked for the received frame.
23 PP	Parity Polarity Controls the polarity of the parity bit transmitted and checked. 0 Even Parity: the number of 1 bits in the transmitted frame is even. The SR[DPEF] bit is set if in the received frame number of 1 bits is odd. 1 Odd Parity: the number of 1 bits in the transmitted frame is odd. The SR[DPEF] bit is set if in the received frame number of 1 bits is even.
24–31 DPCSx	DSI Peripheral Chip Select 0–7 Selects which of the PCS signals to assert during a DSI master mode transfer. 0 Negate PCS[x]. 1 Assert PCS[x].

42.4.13 DSI Serialization Data Register 0 (DSPI_SDR0)

SDR0 contains the states of the 32 LSB Parallel Input signals. The states of the Parallel Input signals are latched into the SDR0 on the rising edge of every bus clock. The SDR0 is read-only. When the TXSS bit in the DSICR0 is cleared, the data in the SDR0 is used as the source of the DSI frames.

Address: 0h base + C0h offset = C0h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	SER_DATA																															
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

DSPI_SDR0 field descriptions

Field	Description
0–31 SER_DATA	Serialized Data Contains the signal states of the 32 LSB Parallel Input signals.

42.4.14 DSI Alternate Serialization Data Register 0 (DSPI_ASDR0)

ASDR0 is used by host software to write the 32 LSB of the data to be serialized. When the TXSS bit in the DSICR0 is set, the data in the ASDR0 is the source of the DSI frames. Writes to the ASDR0 take effect on the next frame boundary.

Address: 0h base + C4h offset = C4h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R																																
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

DSPI_ASDR0 field descriptions

Field	Description
0–31 ASER_DATA	Alternate Serialized Data Holds the alternate 32 LSB of the data to be serialized.

42.4.15 DSI Transmit Comparison Register 0 (DSPI_COMPR0)

COMPR0 holds a copy of the 32 LSB of the last transmitted DSI data. COMPR0 is read-only. DSI data is transferred to this register as it is loaded into the TX Shift Register.

Address: 0h base + C8h offset = C8h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R																																
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

DSPI_COMPR0 field descriptions

Field	Description
0–31 COMP_DATA	Compare Data

DSPI_COMPR0 field descriptions (continued)

Field	Description
	Holds the 32 LSB of the last serialized DSI data.

42.4.16 DSI Deserialization Data Register 0 (DSPI_DDR0)

DDR0 holds the 32 LSB signal states for the Parallel Output signals. DDR0 is read-only and host software can read data from incoming DSI frames.

Address: 0h base + CCh offset = CCh

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	DESER_DATA																															
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

DSPI_DDR0 field descriptions

Field	Description
0–31 DESER_DATA	Deserialized Data Holds 32 LSB of deserialized data that is presented as signal states to the Parallel Output signals.

42.4.17 DSI Configuration Register 1 (DSPI_DSICR1)

Selects various attributes associated with the following configurations:

- DSI
- TSB

The user must not write to the DSICR1 while the module is in the Running state.

Address: 0h base + D0h offset = D0h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0			TSBCNT					0			0	0	0	DSE1	DSE0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31				
R	0								DPCS1_x											
W																				
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0				

DSPI_DSICR1 field descriptions

Field	Description
0–2 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
3–7 TSBCNT	Timed Serial Bus Operation Count When TSB is enabled (DSICR0[TSBC]), TSBCNT defines the length of the data frame. The length value can vary from 3 to 31 (00011b to 11111b). The length value specifies the number of data bits to be shifted out during a transfer in TSB mode. The number of data bits in the data frame is one more than the value in the TSBCNT field.
8–10 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
11 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
12 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
13 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
14 DSE1	Data Select Enable 1 When the TBSC bit is set, the DSE1 bit controls insertion of the zero bit (Data Select) in the middle of the data frame. The insertion bit position is defined by the FMSZ field of CTARn register, selected by the DSICR0[DSICTAS] field. The TSBCNT field value must be greater than the position of insertion for proper operation of DSE1. 0 No Zero bit is inserted in the middle of the data frame. 1 Zero bit is inserted at the middle of the data frame. Total number of bits in the data frame is increased by 1.
15 DSE0	Data Select Enable 0 When the TBSC bit is set, the DSE1 bit controls insertion of the zero bit (Data Select) in the beginning of the DSI frame. 0 No Zero bit is inserted in the beginning of the frame. 1 Zero bit is inserted at the beginning of the data frame. Total number of bits in the data frame is increased by 1.
16–23 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
24–31 DPCS1_x	DSI Peripheral Chip Select 0–7 Specifies the PCSs to assert for the second part of the DSI frame when operating in TSB configuration with dual receiver. DPCS1 selects the PCS signals to assert during the second part of the DSI frame. DPCS1 controls the assertions of PCS signals only in TSB mode. 0 Negate PCS[x]. 1 Assert PCS[x].

42.4.18 DSI Serialization Source Select Register 0 (DSPI_SSR0)

SSR0 is used to create a combined frame for transmission that contains bits from ASDR0 and SDR0. Each bit in the SSR0 register selects a corresponding bit to be serialized. When DSICR0[TXSS] is set, the SSR0 register value has no effect.

Address: 0h base + D4h offset = D4h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R																																
W																																
	SS																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	

DSPI_SSR0 field descriptions

Field	Description
0–31 SS	Source Select Select the serialization source for the 32 LSB of the DSI frame. Each SS bit selects data for a corresponded bit in the transmitted frame. 0 The bit in the transmitted frame is taken from the Parallel Input pin. 1 The bit in the transmitted frame is taken from the ASDR0 register

42.4.19 DSI Parallel Input Select Register 0 (DSPI_PISR0)

PISR0–3 select each data bit for the transmitted frame from 32 parallel input pins. Each Input Pin Select (IPS) field controls one bit in the transmitted frame. Each register contains control fields for 8 bits in the frame. The select field value is defined as a 4-bit signed integer number. The selected parallel input pin number is defined as a difference between the sum of the field number and the field value.

For example, if IPS16 is equal to the binary number 1111 (minus 1 decimal), then bit 16 in the frame is taken from parallel input pin number 15. When the IPS0 is equal to -1, then bit 0 in the frame is taken from parallel input 31. When the IPS0 is equal to +1, then bit 0 in the frame is taken from Parallel Input 1 and so on.

Note that PISR0–3 only preselect the Parallel Input pins. The final selection to the transmitted frame is made with the SSR0-1 register bits or the DSICR0[TXSS] bit.

Memory Map/Register Definition

Address: 0h base + D8h offset = D8h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R																																
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

DSPI_PISR0 field descriptions

Field	Description
0–3 IPS7	Input Pin Select 7 Selects the Parallel Input pin for transmitted frame bit 7.
4–7 IPS6	Input Pin Select 6 Selects the Parallel Input pin for transmitted frame bit 6.
8–11 IPS5	Input Pin Select 5 Selects the Parallel Input pin for transmitted frame bit 5.
12–15 IPS4	Input Pin Select 4 Selects the Parallel Input pin for transmitted frame bit 4.
16–19 IPS3	Input Pin Select 3 Selects the Parallel Input pin for transmitted frame bit 3.
20–23 IPS2	Input Pin Select 2 Selects the Parallel Input pin for transmitted frame bit 2.
24–27 IPS1	Input Pin Select 1 Selects the Parallel Input pin for transmitted frame bit 1.
28–31 IPS0	Input Pin Select 0 Selects the Parallel Input pin for transmitted frame bit 0.

42.4.20 DSI Parallel Input Select Register 1 (DSPI_PISR1)

Address: 0h base + DCh offset = DCh

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R																																
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

DSPI_PISR1 field descriptions

Field	Description
0–3 IPS15	Input Pin Select 15 Selects the Parallel Input pin for transmitted frame bit 15.
4–7 IPS14	Input Pin Select 14

Table continues on the next page...

DSPI_PISR1 field descriptions (continued)

Field	Description
	Selects the Parallel Input pin for transmitted frame bit 14.
8–11 IPS13	Input Pin Select 13 Selects the Parallel Input pin for transmitted frame bit 13.
12–15 IPS12	Input Pin Select 12 Selects the Parallel Input pin for transmitted frame bit 12.
16–19 IPS11	Input Pin Select 11 Selects the Parallel Input pin for transmitted frame bit 11.
20–23 IPS10	Input Pin Select 10 Selects the Parallel Input pin for transmitted frame bit 10.
24–27 IPS9	Input Pin Select 9 Selects the Parallel Input pin for transmitted frame bit 9.
28–31 IPS8	Input Pin Select 8 Selects the Parallel Input pin for transmitted frame bit 8.

42.4.21 DSI Parallel Input Select Register 2 (DSPI_PISR2)

Address: 0h base + E0h offset = E0h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	IPS23				IPS22				IPS21				IPS20				IPS19				IPS18				IPS17				IPS16			
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	

DSPI_PISR2 field descriptions

Field	Description
0–3 IPS23	Input Pin Select 23 Selects the Parallel Input pin for transmitted frame bit 23.
4–7 IPS22	Input Pin Select 22 Selects the Parallel Input pin for transmitted frame bit 22.
8–11 IPS21	Input Pin Select 21 Selects the Parallel Input pin for transmitted frame bit 21.
12–15 IPS20	Input Pin Select 20 Selects the Parallel Input pin for transmitted frame bit 20.
16–19 IPS19	Input Pin Select 19 Selects the Parallel Input pin for transmitted frame bit 19.

Table continues on the next page...

DSPI_PISR2 field descriptions (continued)

Field	Description
20–23 IPS18	Input Pin Select 18 Selects the Parallel Input pin for transmitted frame bit 18.
24–27 IPS17	Input Pin Select 17 Selects the Parallel Input pin for transmitted frame bit 17.
28–31 IPS16	Input Pin Select 16 Selects the Parallel Input pin for transmitted frame bit 16.

42.4.22 DSI Parallel Input Select Register 3 (DSPI_PISR3)

Address: 0h base + E4h offset = E4h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R																																
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

DSPI_PISR3 field descriptions

Field	Description
0–3 IPS31	Input Pin Select 31 Selects the Parallel Input pin for transmitted frame bit 31.
4–7 IPS30	Input Pin Select 30 Selects the Parallel Input pin for transmitted frame bit 30.
8–11 IPS29	Input Pin Select 29 Selects the Parallel Input pin for transmitted frame bit 29.
12–15 IPS28	Input Pin Select 28 Selects the Parallel Input pin for transmitted frame bit 28.
16–19 IPS27	Input Pin Select 27 Selects the Parallel Input pin for transmitted frame bit 27.
20–23 IPS26	Input Pin Select 26 Selects the Parallel Input pin for transmitted frame bit 26.
24–27 IPS25	Input Pin Select 25 Selects the Parallel Input pin for transmitted frame bit 25.
28–31 IPS24	Input Pin Select 24 Selects the Parallel Input pin for transmitted frame bit 24.

42.4.23 DSI Deserialized Data Interrupt Mask Register 0 (DSPI_DIMR0)

DIMR0 selects bits in the 32 least-significant bits of the received DSI frame to be checked to generate the DDI interrupt.

Address: 0h base + E8h offset = E8h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	MASK																															
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

DSPI_DIMR0 field descriptions

Field	Description
0–31 MASK	Mask Defines which bits in the 32 LSB of received deserialization data should be checked to produce the Deserialized Data Interrupt (DDI). 0 The bit in the received DSI frame does not produce a DDI interrupt. 1 The bit in the received DSI frame can produce a DDI interrupt if the data bit matches the configured polarity.

42.4.24 DSI Deserialized Data Polarity Interrupt Register 0 (DSPI_DPIR0)

DPIR0 defines which data bit value in the 32 least-significant bits of the received DSI frame generates the DDI interrupt.

Address: 0h base + ECh offset = ECh

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	DP																															
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

DSPI_DPIR0 field descriptions

Field	Description
0–31 DP	Data Polarity Defines which value of the 32 LSB of received deserialization data sets the SR[DDIF] bit.

DSPI_DPIR0 field descriptions (continued)

Field	Description
0	If the received bit is 0, the SR[DDIF] bit is set.
1	If the received bit is 1, the SR[DDIF] bit is set.

42.5 Functional description

The module supports full-duplex, synchronous serial communications between chips and peripheral devices. The module can also be used to reduce the number of pins required for I/O by serializing and deserializing Parallel Input/Output signals. All communications are done with SPI-like protocol.

The module has the following configurations

- The SPI Configuration in which the module operates as a basic SPI or a queued SPI.
- DSI Configuration in which the module serializes and deserializes Parallel Input/Output signals or bits from memory-mapped registers
- When Master mode is enabled, the module also supports CSI Configuration, which combines the functionality of the SPI and DSI configurations.

The DCONF field in the Module Configuration Register (MCR) determines the module Configuration. See [Module Configuration Register \(DSPI_MCR\)](#) for the DSPI configuration values.

The CTARn registers hold clock and transfer attributes. The SPI configuration allows to select which CTAR to use on a frame by frame basis by setting a field in the SPI command.

The DSI configuration statically selects which CTAR to use. In CSI, Configuration priority logic determines if SPI data or DSI data is transferred and dictates what CTAR register is used for the data transfer. See [Clock and Transfer Attributes Register \(In Master Mode\) \(DSPI_CTARn\)](#) for information on the fields of CTAR registers.

Typical master to slave connections are shown in the following figure. When a data transfer operation is performed, data is serially shifted a predetermined number of bit positions. Because the modules are linked, data is exchanged between the master and the slave. The data that was in the master shift register is now in the shift register of the slave, and vice versa. At the end of a transfer, the Transfer Control Flag(TCF) bit in the Shift Register(SR) is set to indicate a completed frame transfer.

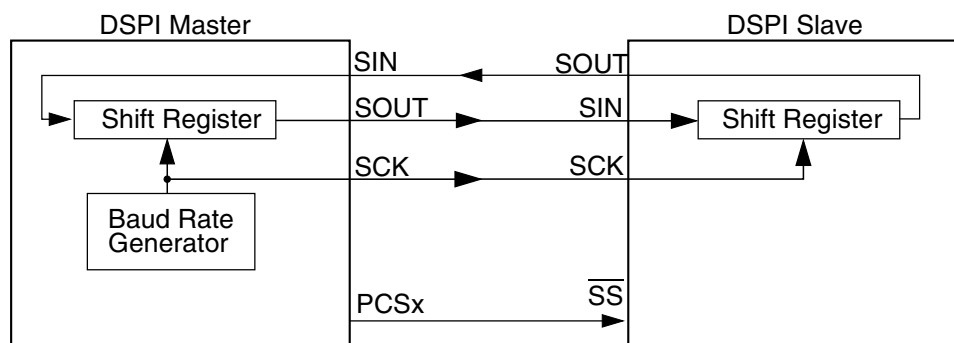


Figure 42-45. Serial protocol overview

Generally, more than one slave device can be connected to the module master. 6 Peripheral Chip Select (PCS) signals of the module masters can be used to select which of the slaves to communicate with. Refer to the chip specific section for details on the number of PCS signals used in this chip.

The three DSPI configurations share transfer protocol and timing properties which are described independently of the configuration in [Transfer formats](#). The transfer rate and delay settings are described in [Module baud rate and clock delay generation](#).

42.5.1 Start and Stop of module transfers

The module has two operating states: Stopped and Running. Both the states are independent of it's configuration. The default state of the module is Stopped. In the Stopped state, no serial transfers are initiated in Master mode and no transfers are responded to in Slave mode. The Stopped state is also a safe state for writing the various configuration registers of the module without causing undetermined results. In the Running state serial transfers take place.

The TXRXS bit in the SR indicates the state of module. The bit is set if the module is in Running state.

The module starts or transitions to Running when all of the following conditions are true:

- SR[EOQF] bit is clear
- Chip is not in the Debug mode or the MCR[FRZ] bit is clear
- MCR[HALT] bit is clear

The module stops or transitions from Running to Stopped after the current frame when any one of the following conditions exist:

- SR[EOQF] bit is set

- Chip in the Debug mode and the MCR[FRZ] bit is set
- MCR[HALT] bit is set

State transitions from Running to Stopped occur on the next frame boundary if a transfer is in progress, or immediately if no transfers are in progress.

42.5.2 Serial Peripheral Interface (SPI) configuration

The SPI configuration transfers data serially using a shift register and a selection of programmable transfer attributes. The module is in SPI configuration when the DCONF field in the MCR is 0b00. The SPI frames can be 32 bits long. The host CPU or a DMA controller transfers the SPI data from the external to the module RAM queues to a TX FIFO buffer. The received data is stored in entries in the RX FIFO buffer. The host CPU or the DMA controller transfers the received data from the RX FIFO to memory external to the module. The operation of FIFO buffers is described in the following sections:

- [Transmit First In First Out \(TX FIFO\) buffering mechanism](#)
- [Command First In First Out \(CMD FIFO\) Buffering Mechanism](#)
- [Receive First In First Out \(RX FIFO\) buffering mechanism](#)

The interrupt and DMA request conditions are described in [Interrupts/DMA requests](#).

The SPI configuration supports two block-specific modes—Master mode and Slave mode. In Master mode the module initiates and controls the transfer according to the fields of the executing SPI Command. In Slave mode, the module responds only to transfers initiated by a bus master external to it and the SPI command field space is reserved.

42.5.2.1 Master mode

In SPI Master mode, the module initiates the serial transfers by controlling the SCK and the PCS signals. The executing SPI Command determines which CTARs will be used to set the transfer attributes and which PCS signals to assert. The command field also contains various bits that help with queue management and transfer protocol. See [PUSH TX FIFO Register In Master Mode \(DSPI_PUSHR\)](#) for details on the SPI command fields. The data in the executing TX FIFO entry is loaded into the shift register and shifted out on the Serial Out (SOUT) pin. In SPI Master mode, each SPI frame to be transmitted has a command associated with it, allowing for transfer attribute control on a frame by frame basis.

42.5.2.2 Slave mode

In SPI Slave mode the module responds to transfers initiated by an SPI bus master. It does not initiate transfers. Certain transfer attributes such as clock polarity, clock phase, and frame size must be set for successful communication with an SPI master. The SPI Slave mode transfer attributes are set in the CTAR0. The data is shifted out with MSB first. Shifting out of LSB is not supported in this mode.

42.5.2.3 FIFO disable operation

The FIFO disable mechanisms allow SPI transfers without using the TX FIFO, CMD FIFO or RX FIFO. The module operates as a double-buffered simplified SPI when the FIFOs are disabled. The Transmit and Receive side of the FIFOs are disabled separately. Setting the MCR[DIS_TXF] bit disables the TX FIFO and CMD FIFO, and setting the MCR[DIS_RXF] bit disables the RX FIFO.

The FIFO disable mechanisms are transparent to the user and to host software. Transmit data and commands are written to the PUSHHR and received data is read from the POPR.

When the TX FIFO and CMD FIFO are disabled:

- SR[TFFF], SR[TFUF] and SR[TXCTR] behave as if there is a one-entry FIFO
- The contents of TXFRs, SR[TXNXTPTR] are undefined

Similarly, when the RX FIFO is disabled, the RFDF, RFOF, and RXCTR fields in the SR behave as if there is a one-entry FIFO, but the contents of the RXFR registers and POPNXTPTR are undefined.

42.5.2.4 Transmit First In First Out (TX FIFO) buffering mechanism

The TX FIFO functions as a buffer of SPI data for transmission. The TX FIFO holds 4 words, each consisting of SPI data. The number of entries in the TX FIFO is device-specific. SPI data is added to the TX FIFO by writing to the Data Field of module PUSH FIFO Register (PUSHHR). TX FIFO entries can only be removed from the TX FIFO by being shifted out or by flushing the TX FIFO.

The TX FIFO Counter field (TXCTR) in the module Status Register (SR) indicates the number of valid entries in the TX FIFO. The TXCTR is updated every time a 8- or 16-bit write takes place to PUSHHR[TXDATA] or SPI data is transferred into the shift register from the TX FIFO.

The TXNXTPTR field indicates the TX FIFO Entry that will be transmitted during the next transfer. The TXNXTPTR field is incremented every time SPI data is transferred from the TX FIFO to the shift register. The maximum value of the field is equal to the maximum implemented TXFR number and it rolls over after reaching the maximum.

42.5.2.4.1 Filling the TX FIFO

Host software or other intelligent blocks can add (push) entries to the TX FIFO and CMD FIFO by writing to the PUSHR. When the TX FIFO is not full, the TX FIFO Fill Flag (TFFF) in the SR is set. The TFFF bit is cleared when TX FIFO is full and the DMA controller indicates that a write to PUSHR is complete. Writing a '1' to the TFFF bit also clears it. The TFFF can generate a DMA request or an interrupt request. See [Transmit FIFO Fill Interrupt or DMA Request](#) for details.

The module ignores attempts to push data to a full TX FIFO, and the state of the TX FIFO does not change and no error condition is indicated.

42.5.2.4.2 Draining the TX FIFO

The TX FIFO entries are removed (drained) by shifting SPI data out through the shift register. Entries are transferred from the TX FIFO to the shift register and shifted out as long as there are valid entries in the TX FIFO. Every time an entry is transferred from the TX FIFO to the shift register, the TX FIFO Counter decrements by one. At the end of a transfer, the TCF bit in the SR is set to indicate the completion of a transfer. The TX FIFO is flushed by writing a '1' to the CLR_TXF bit in MCR.

If an external bus master initiates a transfer with a module slave while the slave's TX FIFO is empty, the Transmit FIFO Underflow Flag (TFUF) in the slave's SR is set. See [Transmit FIFO Underflow Interrupt Request](#) for details.

42.5.2.5 Command First In First Out (CMD FIFO) Buffering Mechanism

The CMD FIFO functions as a buffer of SPI command used for SPI data transmission. The TX FIFO and CMD FIFO must be filled together, i.e. write enables should be given for both the Data and Command fields while performing a PUSHR operation.

The CMD FIFO holds 4 words, each representing SPI command fields. The number of entries in the CMD FIFO is device-specific. SPI Command is added to the CMD FIFO by writing to the command field of DSPI PUSH FIFO Register (PUSHR). CMD FIFO entries can only be removed from the CMD FIFO by being shifted out (to help transmit SPI data) or by flushing the CMD FIFO.

Every CMD FIFO entry has a corresponding single TX FIFO entry attached to it because both these FIFO's are filled simultaneously.

42.5.2.6 Receive First In First Out (RX FIFO) buffering mechanism

The RX FIFO functions as a buffer for data received on the SIN pin. The RX FIFO holds 4 received SPI data frames. The number of entries in the RX FIFO is device-specific. SPI data is added to the RX FIFO at the completion of a transfer when the received data in the shift register is transferred into the RX FIFO. SPI data are removed (popped) from the RX FIFO by reading the module POP RX FIFO Register (POPR). RX FIFO entries can only be removed from the RX FIFO by reading the POPR or by flushing the RX FIFO.

The RX FIFO Counter field (RXCTR) in the module's Status Register (SR) indicates the number of valid entries in the RX FIFO. The RXCTR is updated every time the POPR is read or SPI data is copied from the shift register to the RX FIFO.

The POPNXTPTR field in the SR points to the RX FIFO entry that is returned when the POPR is read. The POPNXTPTR contains the positive offset from RXFR0 in a number of 32-bit registers. For example, POPNXTPTR equal to two means that the RXFR2 contains the received SPI data that will be returned when the POPR is read. The POPNXTPTR field is incremented every time the POPR is read. The maximum value of the field is equal to the maximum implemented RXFR number and it rolls over after reaching the maximum.

42.5.2.6.1 Filling the RX FIFO

The RX FIFO is filled with the received SPI data from the shift register. While the RX FIFO is not full, SPI frames from the shift register are transferred to the RX FIFO. Every time an SPI frame is transferred to the RX FIFO, the RX FIFO Counter is incremented by one.

If the RX FIFO and shift register are full and a transfer is initiated, the RFOF bit in the SR is set indicating an overflow condition. Depending on the state of the ROOE bit in the MCR, the data from the transfer that generated the overflow is either ignored or shifted in to the shift register. If the ROOE bit is set, the incoming data is shifted in to the shift register. If the ROOE bit is cleared, the incoming data is ignored.

42.5.2.6.2 Draining the RX FIFO

Host CPU or a DMA can remove (pop) entries from the RX FIFO by reading the module POP RX FIFO Register (POPR). A read of the POPR decrements the RX FIFO Counter by one. Attempts to pop data from an empty RX FIFO are ignored and the RX FIFO Counter remains unchanged. The data, read from the empty RX FIFO, is undetermined.

When the RX FIFO is not empty, the RX FIFO Drain Flag (RFDF) in the SR is set. The RFDF bit is cleared when the RX_FIFO is empty and the DMA controller indicates that a read from POPR is complete or by writing a 1 to it.

42.5.3 Deserial Serial Interface (DSI) Configuration

The DSI Configuration supports pin count reduction by serializing Parallel Input signals or register bits and shifting them out in a SPI-like protocol. The timing and transfer protocol is described in [Transfer formats](#). The received serial frames are converted to a parallel form (deserialized) and placed on the Parallel Output signals or in the DSPI_DDR register. The various features of the DSI Configuration are set in DSPI DSI Configuration Registers (DSPI_DSICR0/1).

The DSI frames can be from 4 to 32 bits.

42.5.3.1 DSI Master Mode

In DSI master mode the DSPI initiates and controls the DSI transfers. The DSI master has different conditions that can initiate a transfer:

- Continuous
- Change in data
- Trigger signal
- Trigger signal combined with a change in data

These transfer initiation conditions are described in [DSI Transfer Initiation Control](#). Transfer attributes are set during initialization. The DSICTAS field in the DSPI_DSICR0 determines which of the DSPI_CTAR registers will control the transfer attributes.

42.5.3.2 Slave Mode

In DSI slave mode the DSPI responds to transfers initiated by a SPI or DSI bus master. In this mode the DSPI does not initiate DSI transfers. Certain transfer attributes such as clock polarity and phase must be set for successful communication with a DSI master. The DSI slave mode transfer attributes are set in the DSPI_CTAR1. The data is shifted out with MSB first.

42.5.3.3 DSI Serialization

In the DSI Configuration, up to 32 bits can be serialized using two different sources. The TXSS bit in the DSPI_DSICR0 selects between the DSPI DSI Serialization Data Register (DSPI_SDR0) and the DSPI DSI Alternate Serialization Data Register (DSPI_ASDR0) as the source of the serialized data. The DSPI_SDR0 holds the latest 32 Parallel Input signal values which is sampled at every rising edge of the bus clock. The DSPI_ASDR0 register is written by host software and used as an alternate source of serialized data.

The DSPI has to sample the inputs (from SDR0 or ASDR0) before it transmits a DSI frame. Due to the asynchronous clocking structure within DSPI, the sampled data (from SDR0 or ASDR0) must be synchronized into the Protocol Domain before being loaded into the Shift Register for transmission.

To prevent Clock Domain Crossing issues, inputs from SDR0 or ASDR0 should not change while they are being sampled into the Protocol Domain. This is ensured in the following manner for the ASDR0 inputs. The initial write performed on ASDR0 register will cause the 32-bit ASDR0 input to be latched into a temporary register which is then locked (further latching is prevented) until this data gets synchronized into the Protocol Domain. Once synchronized, the lock is removed and the next ASDR0 write will again trigger this process. If there was a write performed on ASDR0 while the lock was asserted, this will be taken into consideration as soon as the previous synchronization process completes and lock gets de-asserted. In case, multiple writes were performed on ASDR0 when lock was asserted, only the latest value will be considered for synchronization once the lock gets de-asserted.

The entire process explained above is also true for the SDR0 input, the only difference being that SDR0 inputs are delayed internally by a single Bus clock (register interface clock) to detect a change in the parallel input pins.

Once the synchronization process for a particular De-serial input begins, it takes up to 5 Bus clock (register interface clock) periods and 5 Protocol clock periods before the synchronization process for the next De-serial input can begin. Hence the writes performed on ASDR0 and SDR0 are constrained with this timing.

The DSPI_PISR0-3 registers allow to change relative position of the Parallel input pins in the transmitted frame. Each transmitted frame bit can be selected from 16 adjacent Parallel Inputs by writing IPSn fields. The IPSn field is treated as 4 bit integer number, representing numbers from -8 to 7. The Parallel Input pin number, selected by IPSn field is defined by the difference between the IPSn field number (n) and the IPSn field value. If the operation result is negative the number 32 should be added. If the result is higher than 32, 32 should be subtracted from the result.

For example, IPS0, set to minus 1 (binary 1111), preselects Parallel Input 1 to 0 position in the transmitted frame.

IPS6, set to 3 (binary 0011), preselects Parallel Input 3 to be bit number 6 in the transmitted frame. The value minus 2 (1110) preselects Parallel Input 8.

IPS31, set to minus 8 (binary 1000), preselects Parallel Input 7 to be bit number 31 in the transmitted frame.

Note

The Parallel Input pin state, to be transmitted, should be selected by TXSS or DSPI_SSR0 register and the frame size should be higher than the bit position in the preselected frame.

DSPI_SSR0 register provides additional way to create the frame for transmission. Each bit from this register is OR'd with the TXSS bit and controls individual transmitted bit source. This way, the transmitted frame can have any combination of the DSPI_SDR0 and DSPI_ASDR0 bits. This feature allows control SPI based devices, requiring control and data fields in the frame. Control field may come from DSPI_ASDR0 register, set by the device's CPU, while data field can be generated by device peripheral modules, like PWM timers.

A copy of the last DSI frame shifted out of the Shift Register is stored in the DSPI DSI Transmit Comparison Register (DSPI_COMPR0). This register provides added visibility for debugging and it serves as a reference for transfer initiation control. The following figure shows the DSI Serialization logic.

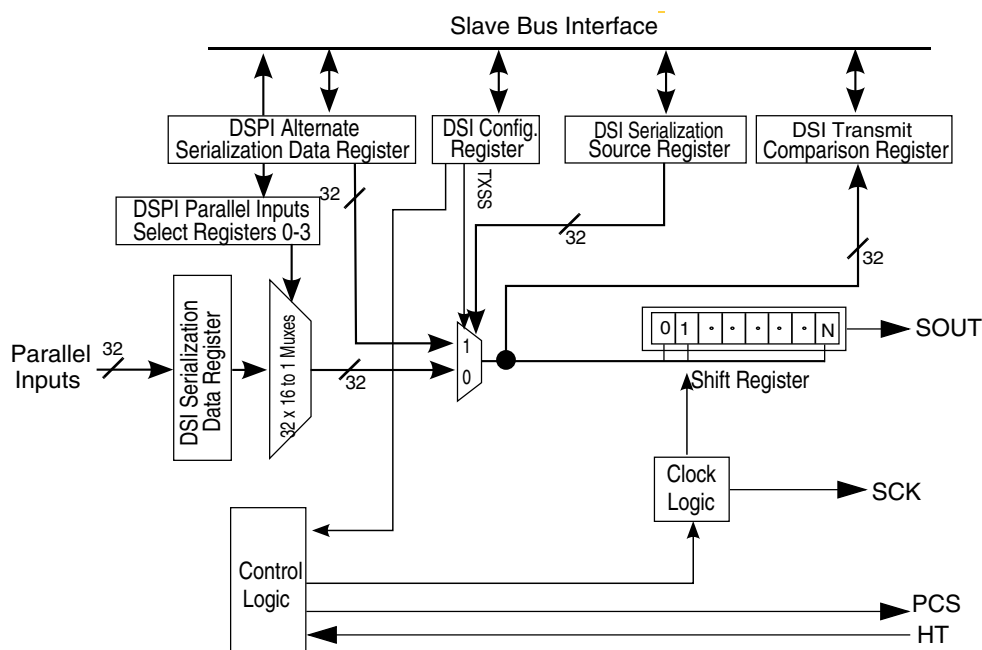


Figure 42-46. DSI Serialization Diagram

42.5.3.4 DSI Deserialization

When all bits in a DSI frame have been shifted in, the frame is copied to the DSPI DSI Deserialization Data Register (DSPI_DDR0). This register presents the deserialized data as Parallel Output signal values. DSPI_DDR0 is memory mapped to allow host software to read the deserialized data directly.

The received data is bit-wise compared to the value of the DSI Deserialized Data Polarity Interrupt Register (DSPI_DPIR0), bit-wise AND'ed with DSI Deserialized Interrupt Mask Register (DSPI_DIMR0) and the results OR'ed to produce DDIF flag in the DSPI_SR register; which in turn can cause DDI interrupt request or DMA request based on RSER[DDIF_RE] and RSER[DDIF_DIRS] bits and/or stop DSI frame transmissions if DSICR0[DMS] bit is set.

The following figure shows the DSI Deserialization logic.

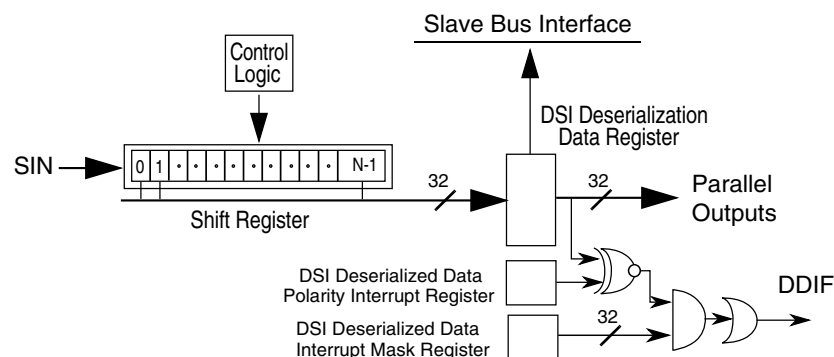


Figure 42-47. DSI Deserialization Diagram

42.5.3.5 DSI Transfer Initiation Control

Data transfers for a DSPI in DSI configuration are initiated by a condition. The transfer initiation conditions are selected by the TRRE and CID bits in the DSPI_DSICR0. The following table lists the four transfer initiation conditions.

Table 42-75. DSI Data Transfer Initiation Control

DSPI_DSICR0 Bits		Transfer Initiation Control
TRRE	CID	
0	0	Continuous
0	1	Change in Data
1	0	Triggered
1	1	Triggered or Change in Data

42.5.3.5.1 Continuous Control

For Continuous Control a new DSI frame shifts out when the previous transfer cycle has completed and the Delay after Transfer (t_{DT}) has elapsed. This has no effect when DSPI is in Slave Mode.

42.5.3.5.2 Change In Data Control

For change in data control a transfer is initiated when the data to be serialized has changed since the transfer of the last DSI frame. A copy of the previously transferred DSI data is stored in DSPI_COMPR0. When the data selected for the transfer from the DSPI_SDR0 and DSPI_ASDR0 registers is different from the data in DSPI_COMPR0 a new DSI frame is transmitted.

NOTE

It is not advisable to configure a DSPI slave with the combination DSICR0[DCONT] = 1, DSICR0[CID] = 1 ; since the following statement then holds true.

Once slave select is asserted, there is a very small window (of up to t module clocks) after every SOF (Start of Frame) during which if a write on ASDR/SDR registers occur, the next frame to be transmitted would be loaded into the shift register. However, it is not feasible that such a write be initiated and completed within this short span of time; hence the data transmitted by DSPI slave would lag by one data frame.

42.5.3.5.3 Triggered Control

For Triggered Control initiation of a transfer is controlled by the Hardware Trigger signal (HT). The TPOL bit in the DSPI_DSICR0 selects the active edge of HT. For HT to have any affect, the TRRE bit in the DSPI_DSICR0 must be set.

42.5.3.5.4 Triggered or Change In Data Control

For Triggered or Change in Data Control, initiation of a transfer is controlled by the HT signal or by the detection of a change in data to be serialized.

42.5.4 Combined Serial Interface (CSI) Configuration

The CSI Configuration of the DSPI is used to support SPI and DSI functions on a frame by frame basis. CSI Configuration allows interleaving of DSI data frames from the parallel input signals with SPI data frames from the TX FIFO. The data returned from the bus slave is either used to drive the Parallel Output signals or it is stored in the RX FIFO. The CSI Configuration allows serialized data and configuration or diagnostic data to be transferred to a slave device using only one serial link. The DSPI is in CSI Configuration when the DCONF field in the DSPI_MCR is 0b10. The following figure shows an example of how a DSPI can be used with a deserializing peripheral that supports SPI control for control and diagnostic frames.

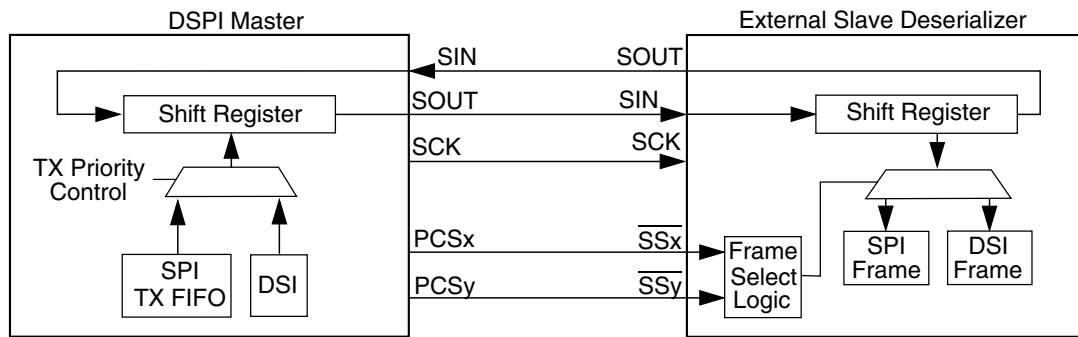


Figure 42-48. Example of System using DSPI in CSI Configuration

In CSI Configuration the DSPI transfers DSI data based on [DSI Transfer Initiation Control](#).

When SPI commands are in the TX FIFO, the SPI frames have priority over the DSI frames. When the TX FIFO or CMD FIFO is empty, DSI transfer resumes.

Two peripheral chip select signals indicate whether DSI data or SPI data is transmitted. The user must configure the DSPI so that the two CTAR registers associated with DSI data and SPI data assert different peripheral chip select signals denoted in the figure as PCSx and PCSy. The CSI Configuration is only supported in master mode.

Data returned from the external slave while a DSI frame is transferred is placed on the Parallel Output signals. Data returned from the external slave while a SPI frame is transferred is moved to the RX FIFO. The TX FIFO, CMD FIFO and RX FIFO are fully functional in CSI mode.

42.5.4.1 CSI Serialization

Serialization in the CSI configuration is similar to serialization in DSI Configuration. The transfer attributes for SPI frames are determined by the DSPI_CTAR register selected by the CTAS field in the SPI command halfword. The transfer attributes for the DSI frames are determined by the DSPI_CTAR register selected by the DSICTAS field in the DSPI_DSICR0.

The Parallel Inputs signal states are latched into the DSPI DSI Serialization Data Register (DSPI_SDR0) and serialized based on the transfer initiation control settings in the DSPI_DSICR0. When SPI frames are written to the TX FIFO (and when CMD FIFO is not empty) they have priority over DSI data from the DSPI_SDR and are transferred at the next frame boundary. A copy of the most recently transferred DSI frame is stored in DSPI_COMPR0. The Transfer Priority Logic selects the source of the serialized data and asserts the appropriate PCS signal.

Separate frame completion interrupts are available to indicate frame transmission completion from SPI and DSI frames.

42.5.4.2 CSI Deserialization

The deserialized frames in CSI Configuration go into the DSPI_DDR0 or the RX FIFO based on the transfer priority logic. When DSI frames are transferred the returned frames are deserialized and latched into the DSPI_DDR0. When SPI frames are transferred the returned frames are deserialized and written to the RX FIFO.

42.5.5 Module baud rate and clock delay generation

The SCK frequency and the delay values for serial transfer are generated by dividing the protocol clock frequency by a prescaler and a scaler with the option for doubling the baud rate. The following figure shows conceptually how the SCK signal is generated.

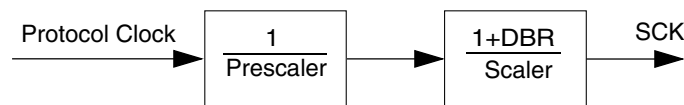


Figure 42-49. Communications clock prescalers and scalers

42.5.5.1 Baud rate generator

The baud rate is the frequency of the SCK. The protocol clock is divided by a prescaler (PBR) and scaler (BR) to produce SCK with the possibility of halving the scaler division. The DBR, PBR, and BR fields in the CTARs select the frequency of SCK by the formula in the BR field description. The following table shows an example of how to compute the baud rate.

Table 42-76. Baud rate computation example

f_p	PBR	Prescaler	BR	Scaler	DBR	Baud rate
100 MHz	0b00	2	0b0000	2	0	25 Mb/s
20 MHz	0b00	2	0b0000	2	1	10 Mb/s

NOTE

The clock frequencies mentioned in the preceding table are given as an example. Refer to the clocking chapter for the frequency used to drive this module in the device.

42.5.5.2 PCS to SCK Delay (t_{csc})

The PCS to SCK delay is the length of time from assertion of the PCS signal to the first SCK edge. See [Figure 42-51](#) for an illustration of the PCS to SCK delay. The PCSSCK and CSSCK fields in the CTAR_x registers select the PCS to SCK delay by the formula in the CSSCK field description. The following table shows an example of how to compute the PCS to SCK delay.

Table 42-77. PCS to SCK delay computation example

f_{sys}	PCSSCK	Prescaler	CSSCK	Scaler	PCS to SCK Delay
100 MHz	0b01	3	0b0100	32	0.96 μ s

PCSSCK and CSSCK fields have no effect in TSB configuration.

NOTE

The clock frequency mentioned in the preceding table is given as an example. Refer to the clocking chapter for the frequency used to drive this module in the device.

42.5.5.3 After SCK Delay (t_{Asc})

The After SCK Delay is the length of time between the last edge of SCK and the negation of PCS. See [Figure 42-51](#) and [Figure 42-52](#) for illustrations of the After SCK delay. The PASC and ASC fields in the CTAR_x registers select the After SCK Delay by the formula in the ASC field description. The following table shows an example of how to compute the After SCK delay.

Table 42-78. After SCK Delay computation example

f_p	PASC	Prescaler	ASC	Scaler	After SCK Delay
100 MHz	0b01	3	0b0100	32	0.96 μ s

PASC and ASC fields have no effect in TSB configuration.

NOTE

The clock frequency mentioned in the preceding table is given as an example. Refer to the clocking chapter for the frequency used to drive this module in the device.

42.5.5.4 Delay after Transfer (t_{DT})

The Delay after Transfer is the minimum time between negation of the PCS signal for a frame and the assertion of the PCS signal for the next frame. See [Figure 42-51](#) for an illustration of the Delay after Transfer. The PDT and DT fields in the CTARx registers select the Delay after Transfer by the formula in the DT field description. The following table shows an example of how to compute the Delay after Transfer.

Table 42-79. Delay after Transfer computation example

f_p	PDT	Prescaler	DT	Scaler	Delay after Transfer
100 MHz	0b01	3	0b1110	32768	0.98 ms

NOTE

The clock frequency mentioned in the preceding table is given as an example. Refer to the clocking chapter for the frequency used to drive this module in the device.

When in Non-Continuous Clock mode the t_{DT} delay is configured according to the equation specified in the CTAR[DT] field description. When in Continuous Clock mode and TSB is not enabled, the delay is fixed at 1 SCK period.

In TSB mode the Delay after Transfer is equal to a number formed by concatenation of PDT and DT fields plus 1 of the SCK clock periods. See detailed information on [Timed Serial Bus \(TSB\)](#)".

42.5.5.5 Peripheral Chip Select Strobe Enable ($\overline{PCSS}$)

The $\overline{PCSS}$ signal provides a delay to allow the PCS signals to settle after a transition occurs thereby avoiding glitches. When the Module is in Master mode and the PCSSE bit is set in the MCR, $\overline{PCSS}$ provides a signal for an external demultiplexer to decode peripheral chip selects other than PCS5 into glitch-free PCS signals. The following figure shows the timing of the $\overline{PCSS}$ signal relative to PCS signals.

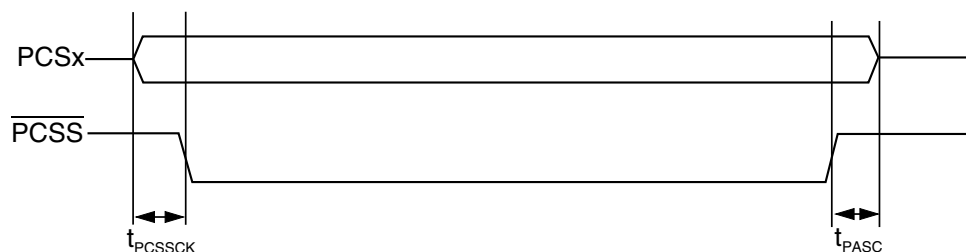


Figure 42-50. Peripheral Chip Select Strobe timing

The delay between the assertion of the PCS signals and the assertion of $\overline{\text{PCSS}}$ is selected by the PCSSCK field in the CTAR based on the following formula:

$$t_{\text{PCSSCK}} = \frac{1}{f_{\text{P}}} \times \text{PCSSCK}$$

At the end of the transfer, the delay between $\overline{\text{PCSS}}$ negation and PCS negation is selected by the PASC field in the CTAR based on the following formula:

$$t_{\text{PASC}} = \frac{1}{f_{\text{P}}} \times \text{PASC}$$

The following table shows an example of how to compute the t_{pcssck} delay.

Table 42-80. Peripheral Chip Select Strobe Assert computation example

f_{P}	PCSSCK	Prescaler	Delay before Transfer
100 MHz	0b11	7	70.0 ns

The following table shows an example of how to compute the t_{pasc} delay.

Table 42-81. Peripheral Chip Select Strobe Negate computation example

f_{P}	PASC	Prescaler	Delay after Transfer
100 MHz	0b11	7	70.0 ns

The $\overline{\text{PCSS}}$ signal is not supported when Continuous Serial Communication SCK or TSB mode is enabled.

NOTE

The clock frequency mentioned in the preceding tables is given as an example. Refer to the clocking chapter for the frequency used to drive this module in the device.

42.5.6 Transfer formats

The SPI serial communication is controlled by the Serial Communications Clock (SCK) signal and the PCS signals. The SCK signal provided by the master device synchronizes shifting and sampling of the data on the SIN and SOUT pins. The PCS signals serve as enable signals for the slave devices.

In Master mode, the CPOL and CPHA bits in the Clock and Transfer Attributes Registers (CTARn) select the polarity and phase of the serial clock, SCK.

- CPOL - Selects the idle state polarity of the SCK
- CPHA - Selects if the data on SOUT is valid before or on the first SCK edge

Even though the bus slave does not control the SCK signal, in Slave mode the values of CPOL and CPHA must be identical to the master device settings to ensure proper transmission. In SPI Slave mode, only CTAR0 is used. In DSI slave mode, only CTAR1 is used.

The module supports four different transfer formats:

- Classic SPI with CPHA=0
- Classic SPI with CPHA=1
- Modified Transfer Format with CPHA = 0
- Modified Transfer Format with CPHA = 1

A modified transfer format is supported to allow for high-speed communication with peripherals that require longer setup times. The module can sample the incoming data later than halfway through the cycle to give the peripheral more setup time. The MTFE bit in the MCR selects between Classic SPI Format and Modified Transfer Format.

In the interface configurations, the module provides the option of keeping the PCS signals asserted between frames. See [Continuous Selection Format](#) for details.

42.5.6.1 Classic SPI Transfer Format (CPHA = 0)

The transfer format shown in following figure is used to communicate with peripheral SPI slave devices where the first data bit is available on the first clock edge. In this format, the master and slave sample their SIN pins on the odd-numbered SCK edges and change the data on their SOUT pins on the even-numbered SCK edges.

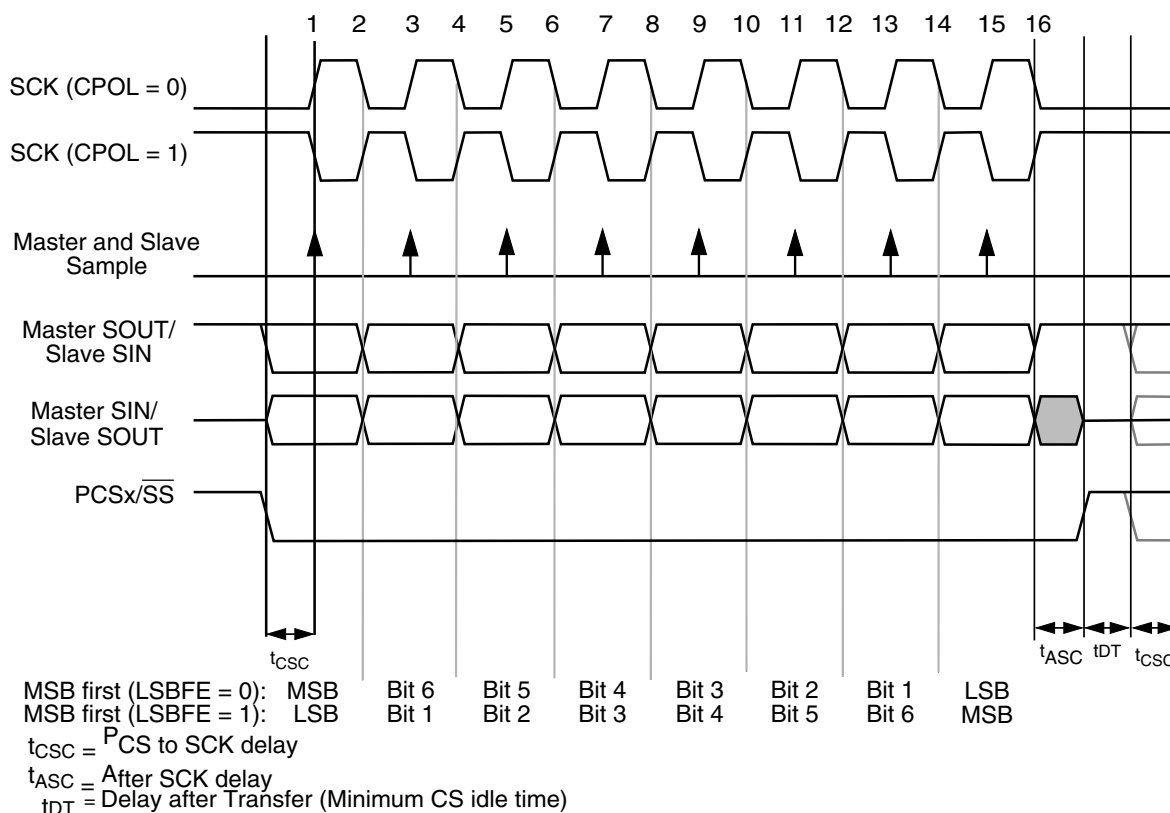


Figure 42-51. Module transfer timing diagram (MTFE=0, CPHA=0, FMSZ=8)

The master initiates the transfer by placing its first data bit on the SOUT pin and asserting the appropriate peripheral chip select signals to the slave device. The slave responds by placing its first data bit on its SOUT pin. After the t_{CSC} delay elapses, the master outputs the first edge of SCK. The master and slave devices use this edge to sample the first input data bit on their serial data input signals. At the second edge of the SCK, the master and slave devices place their second data bit on their serial data output signals. For the rest of the frame the master and the slave sample their SIN pins on the odd-numbered clock edges and changes the data on their SOUT pins on the even-numbered clock edges. After the last clock edge occurs, a delay of t_{ASC} is inserted before the master negates the PCS signals. A delay of t_{DT} is inserted before a new frame transfer can be initiated by the master.

42.5.6.2 Classic SPI Transfer Format (CPHA = 1)

This transfer format shown in the following figure is used to communicate with peripheral SPI slave devices that require the first SCK edge before the first data bit becomes available on the slave SOUT pin. In this format, the master and slave devices change the data on their SOUT pins on the odd-numbered SCK edges and sample the data on their SIN pins on the even-numbered SCK edges.

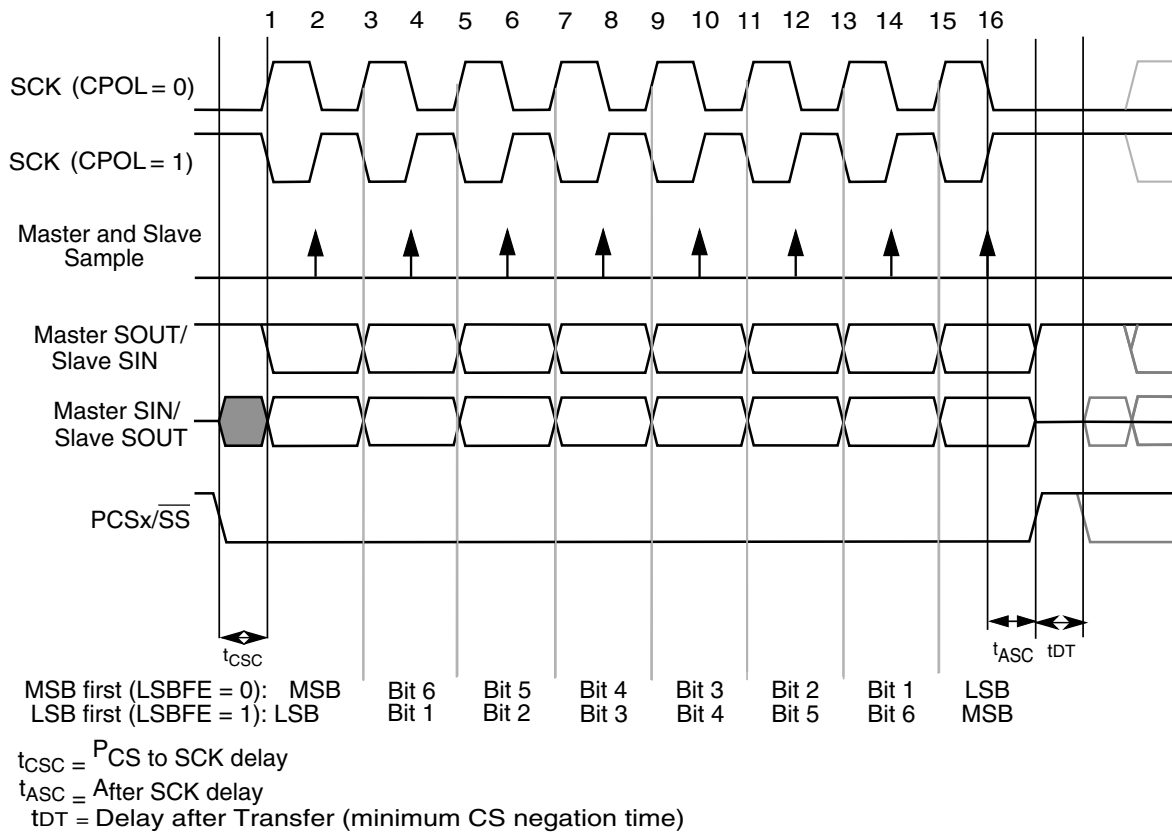


Figure 42-52. Module transfer timing diagram (MTFE=0, CPHA=1, FMSZ=8)

The master initiates the transfer by asserting the PCS signal to the slave. After the t_{CSC} delay has elapsed, the master generates the first SCK edge and at the same time places valid data on the master SOUT pin. The slave responds to the first SCK edge by placing its first data bit on its slave SOUT pin.

At the second edge of the SCK the master and slave sample their SIN pins. For the rest of the frame the master and the slave change the data on their SOUT pins on the odd-numbered clock edges and sample their SIN pins on the even-numbered clock edges. After the last clock edge occurs, a delay of t_{ASC} is inserted before the master negates the PCS signal. A delay of t_{DT} is inserted before a new frame transfer can be initiated by the master.

42.5.6.3 Modified SPI/DSI Transfer Format (MTFE = 1, CPHA = 0)

In this Modified Transfer Format both the master and the slave sample later in the SCK period than in Classic SPI mode to allow the logic to tolerate more delays in device pads and board traces. These delays become a more significant fraction of the SCK period as the SCK period decreases with increasing baud rates.

The master and the slave place data on the SOUT pins at the assertion of the PCS signal. After the PCS to SCK delay has elapsed the first SCK edge is generated. The slave samples the master SOUT signal on every odd numbered SCK edge. The DSPI in the slave mode when the MTFE bit is set also places new data on the slave SOUT on every odd numbered clock edge. Regular external slave, configured with CPHA=0 format drives its SOUT output at every even numbered SCK clock edge.

The DSPI master places its second data bit on the SOUT line one protocol clock after odd numbered SCK edge if the protocol clock frequency to SCK frequency ratio is higher than three. If this ratio is below four the master changes SOUT at odd numbered SCK edge. The point where the master samples the SIN is selected by the DSPI_MCR[SMPL_PT] field. The master sample point can be delayed by one or two protocol clock cycles. The SMPL_PT field should be set to 0 if the protocol to SCK frequency ratio is less than 4. However if this ratio is less than 4, the actual sample point is delayed by one protocol clock cycle automatically by the design.

The following timing diagrams illustrate the DSPI operation with MTFE=1. Timing delays shown are:

- T_{csc} - PCS to SCK assertion delay
- T_{acs} - After SCK PCS negation delay
- $T_{su_{ms}}$ - master SIN setup time
- $T_{hd_{ms}}$ - master SIN hold time
- $T_{vd_{sl}}$ - slave data output valid time, time between slave data output SCK driving edge and data becomes valid.
- $T_{su_{sl}}$ - data setup time on slave data input
- $T_{hd_{sl}}$ - data hold time on slave data input
- T_{sys} - protocol clock period.

The following figure shows the modified transfer format for CPHA = 0 and Fsys/Fsck = 4. Only the condition where CPOL = 0 is illustrated. Solid triangles show the data sampling clock edges. The two possible slave behavior are shown.

- Signal, marked "SOUT of Ext Slave", presents regular SPI slave serial output.
- Signal, marked "SOUT of DSPI Slave", presents DSPI in the slave mode with MTFE bit set.

Other MTFE = 1 diagrams show DSPI SIN input as being driven by a regular external SPI slave, configured according DSPI master CPHA programming.

Note

In the following diagrams, f_{sys} represents the protocol clock frequency from which the Baud frequency f_{sck} is derived.

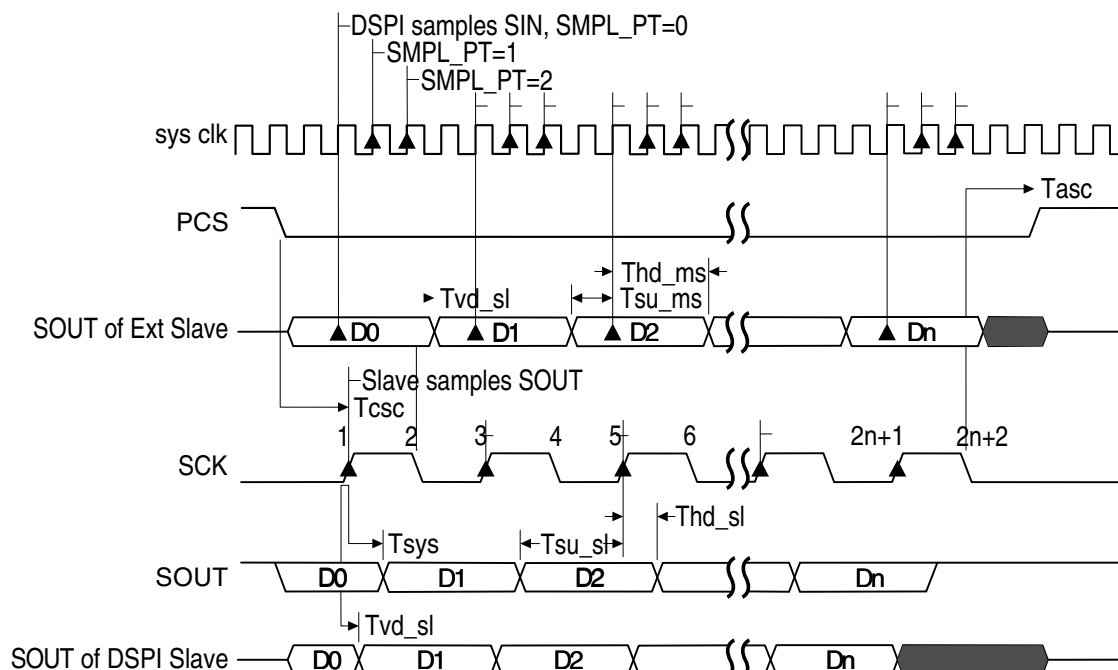


Figure 42-53. DSPI Modified Transfer Format (MTFE=1, CPHA=0, $f_{\text{sck}} = f_{\text{sys}}/4$)

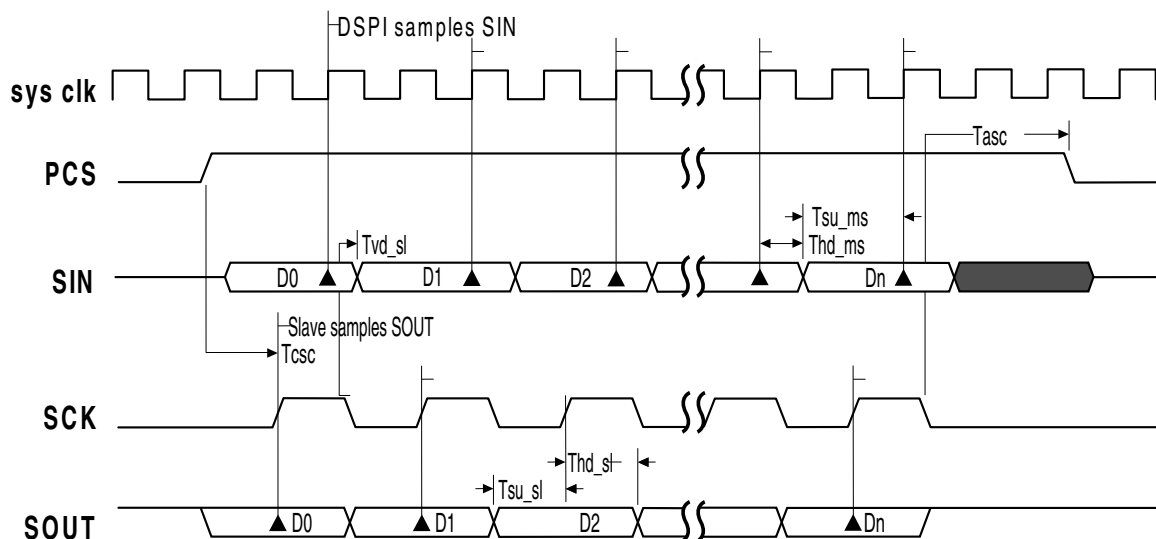


Figure 42-54. DSPI Modified Transfer Format (MTFE=1, CPHA=0, $f_{\text{sck}} = f_{\text{sys}}/2$)

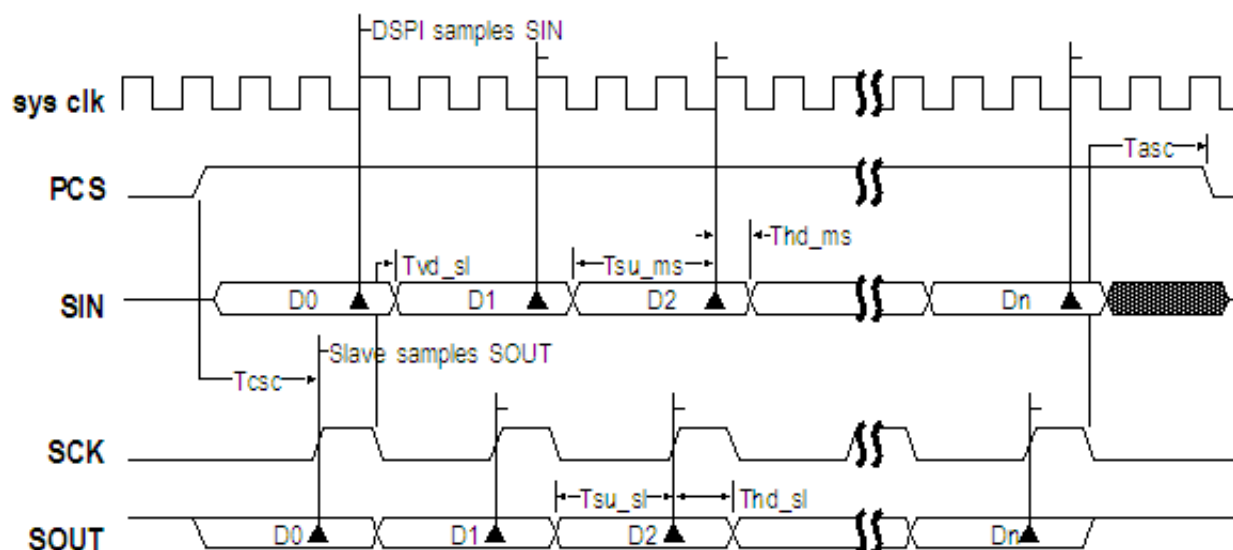


Figure 42-55. DSPI Modified Transfer Format (MTFE=1, CPHA=0, $f_{sck} = f_{sys}/3$)

42.5.6.4 Modified SPI/DSI Transfer Format (MTFE = 1, CPHA = 1)

The following figures show the Modified Transfer Format for CPHA = 1. Only the condition, where CPOL = 0 is shown. At the start of a transfer the DSPI asserts the PCS signal to the slave device. After the PCS to SCK delay has elapsed the master and the slave put data on their SOUT pins at the first edge of SCK. The slave samples the master SOUT signal on the even numbered edges of SCK. The master samples the slave SOUT signal on the odd numbered SCK edges starting with the third SCK edge. The slave samples the last bit on the last edge of the SCK. The master samples the last slave SOUT bit one half SCK cycle after the last edge of SCK. No clock edge will be visible on the master SCK pin during the sampling of the last bit. **The SCK to PCS delay and the After SCK delay must be greater or equal to half of the SCK period.**

NOTE

When MTFE=1 with continuous SCK enabled (MCR [CONT_SCKE] =1) in master mode, configure CTAR[LSBFE]=0 for correct operations while receiving unequal length frames. If PUSHR[CONT] is also set for back to back frame transfer, also configure the frame size of the first frame as less than or equal to the frame size of the next frame. In this scenario, make sure that for all received frames, the bits are read equal to their respective frame sizes and any extra bits during POP operation are masked.

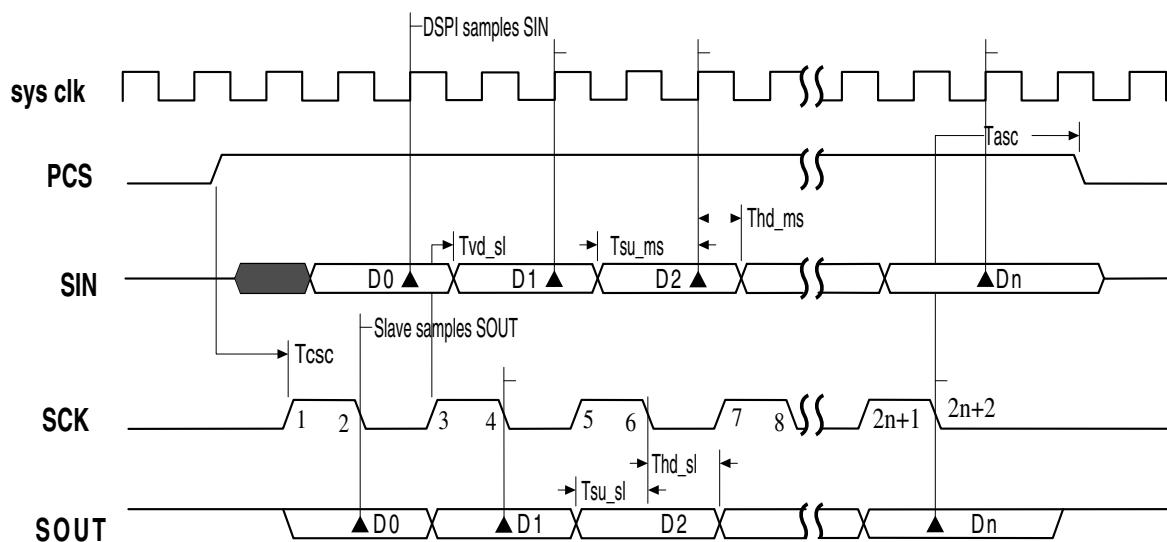


Figure 42-56. DSPI Modified Transfer Format (MTFE=1, CPHA=1, $f_{sck} = f_{sys}/2$)

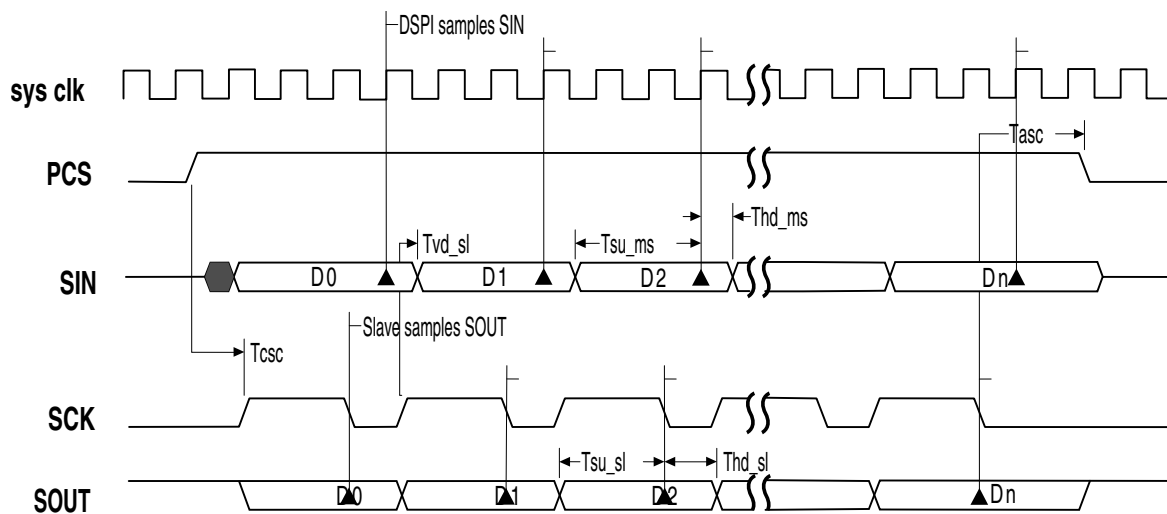


Figure 42-57. DSPI Modified Transfer Format (MTFE=1, CPHA=1, $f_{sck} = f_{sys}/3$)

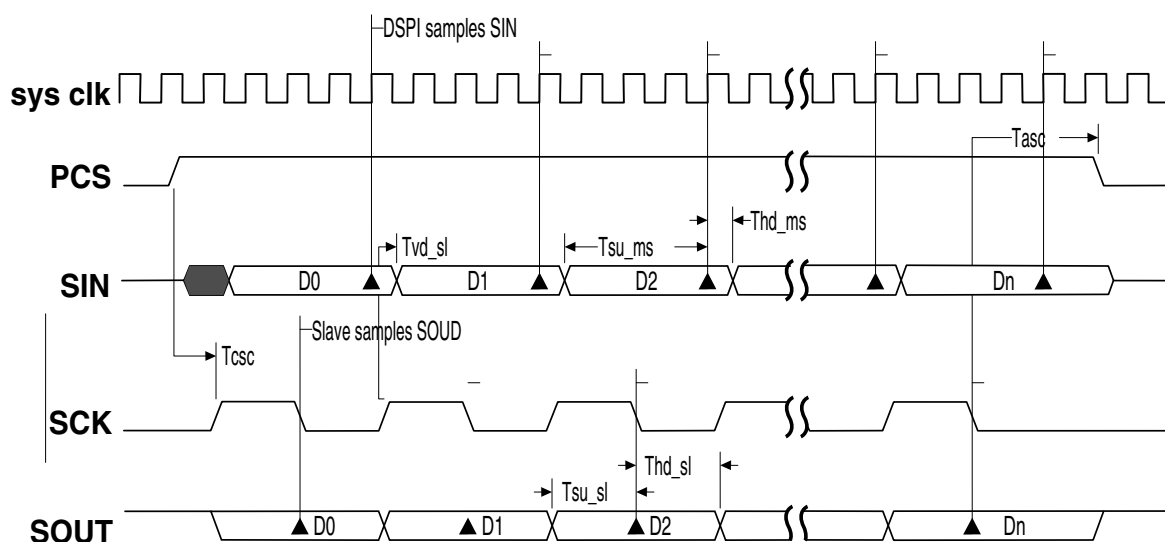


Figure 42-58. DSPI Modified Transfer Format (MTFE=1, CPHA=1, $f_{sck} = f_{sys}/4$)

42.5.6.5 Continuous Selection Format

Some peripherals must be deselected between every transfer. Other peripherals must remain selected between several sequential serial transfers. The Continuous Selection Format provides the flexibility to handle both the following cases. The Continuous Selection Format is enabled for the SPI configuration by setting the CONT bit in the SPI command. Continuous Selection is enabled for the DSI Configuration by setting the DCONT bit in the DSICR0. The behavior of the PCS signals in the two configurations is identical so only SPI configuration will be described.

When the CONT bit = 0, the module drives the asserted Chip Select signals to their idle states in between frames. The idle states of the Chip Select signals are selected by the PCSISn bits in the MCR. The following timing diagram is for two four-bit transfers with CPHA = 1 and CONT = 0.

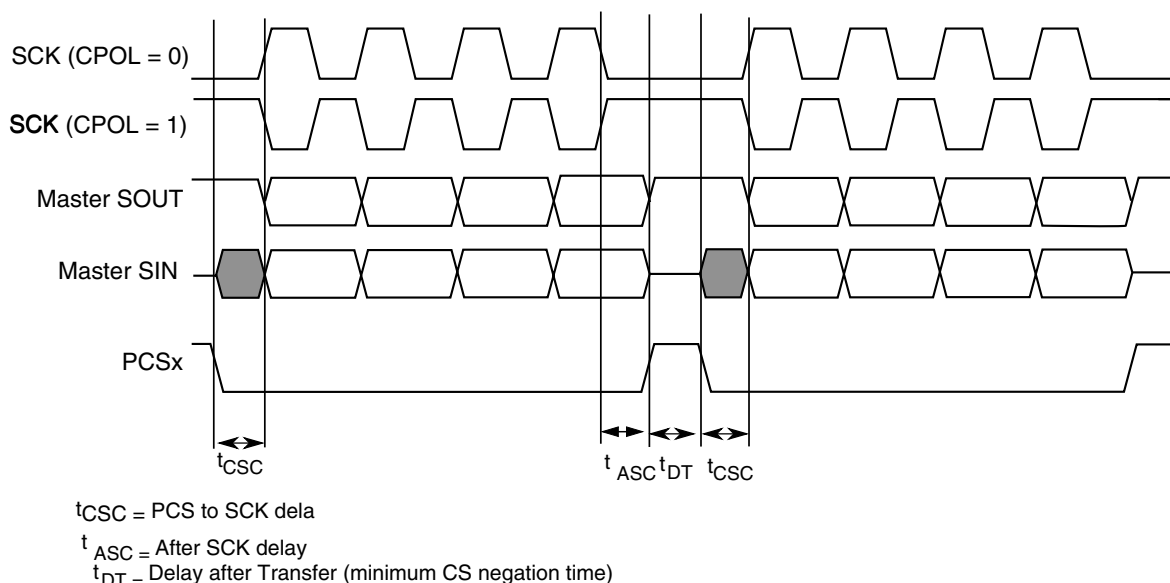


Figure 42-59. Example of non-continuous format (CPHA=1, CONT=0)

When the CONT bit = 1, the PCS signal remains asserted for the duration of the two transfers. The Delay between Transfers (t_{DT}) is not inserted between the transfers. The following figure shows the timing diagram for two four-bit transfers with CPHA = 1 and CONT = 1.

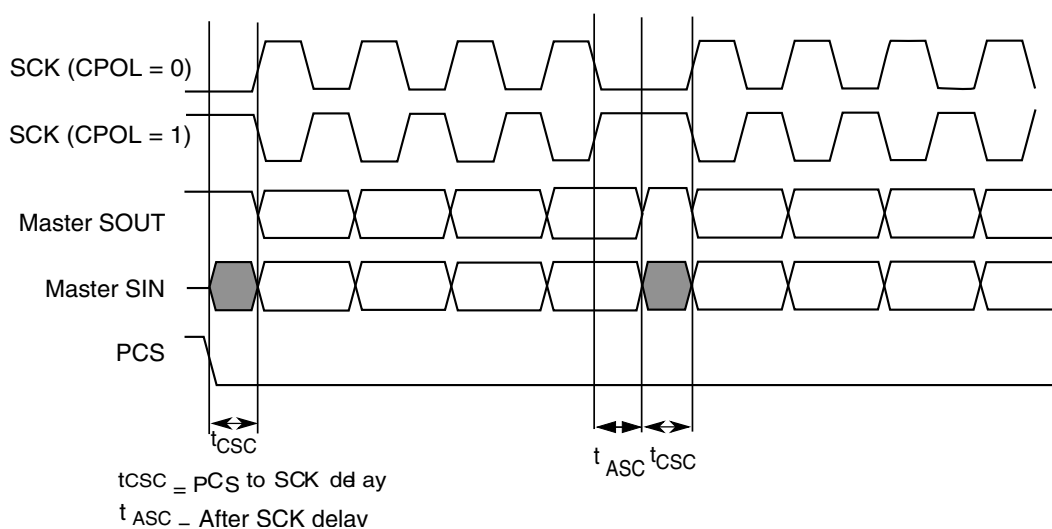


Figure 42-60. Example of continuous transfer (CPHA=1, CONT=1)

When using the module with continuous selection follow these rules:

- All transmit commands must have the same PCSn bits programming.
- The CTARs, selected by transmit commands, must be programmed with the same transfer attributes. Only FMSZ field can be programmed differently in these CTARs.

- When transmitting multiple frames in this mode, the user software must ensure that the last frame has the PUSHHR[CONT] bit deasserted in Master mode and the user software must provide sufficient frames in the TX_FIFO to be sent out in Slave mode and the master deasserts the PCSn at end of transmission of the last frame.
- PUSHHR[CONT] and DSICR0[DCONT] must be deasserted before asserting MCR[HALT] in master mode. This will make sure that the PCSn signals are deasserted. Asserting MCR[HALT] during continuous transfer will cause the PCSn signals to remain asserted and hence Slave Device cannot transition from Running to Stopped state.

NOTE

User must fill the TX FIFO with the number of entries that will be concatenated together under one PCS assertion for both master and slave before the TX FIFO becomes empty.

When operating in Slave mode, ensure that when the last entry in the TX FIFO is completely transmitted, that is, the corresponding TCF flag is asserted and TXFIFO is empty, the slave is deselected for any further serial communication; otherwise, an underflow error occurs.

42.5.7 Continuous Serial Communications Clock

The module provides the option of generating a Continuous SCK signal for slave peripherals that require a continuous clock.

Continuous SCK is enabled by setting the CONT_SCKE bit in the MCR. Enabling this bit generates the Continuous SCK only if MCR[HALT] bit is low. Continuous SCK is valid in all configurations.

Continuous SCK is only supported for CPHA=1. Clearing CPHA is ignored if the CONT_SCKE bit is set. Continuous SCK is supported for Modified Transfer Format.

Clock and transfer attributes for the Continuous SCK mode are set according to the following rules:

- When the module is in SPI configuration, CTAR0 is used initially. At the start of each SPI frame transfer, the CTAR specified by the CTAS for the frame is used.
- When the module is in DSI configuration, the CTAR specified by the DSICTAS field is used at all times.

- When the module is in CSI configuration, the CTAR selected by the DSICTAS field is used initially.
 - At the start of a SPI frame transfer, the CTAR specified by the CTAS value for the frame is used.
 - At the start of a DSI frame transfer, the CTAR specified by the DSICTAS field is used.
- In all configurations, the currently selected CTAR remains in use until the start of a frame with a different CTAR specified, or the Continuous SCK mode is terminated.

It is recommended to keep the baud rate the same while using the Continuous SCK. Switching clock polarity between frames while using Continuous SCK can cause errors in the transfer. Continuous SCK operation is not guaranteed if the module is put into the External Stop mode or Module Disable mode.

Enabling Continuous SCK disables the PCS to SCK delay and the Delay after Transfer (t_{DT}) is fixed to one SCK cycle. When TSB configuration is enabled the t_{DT} is programmable from 1 to 64 SCK cycles. The following figure is the timing diagram for Continuous SCK format with Continuous Selection disabled.

NOTE

In Continuous SCK mode, for the SPI transfer CTAR0 should always be used, and the TX FIFO must be cleared using the MCR[CLR_TXF] field before initiating transfer.

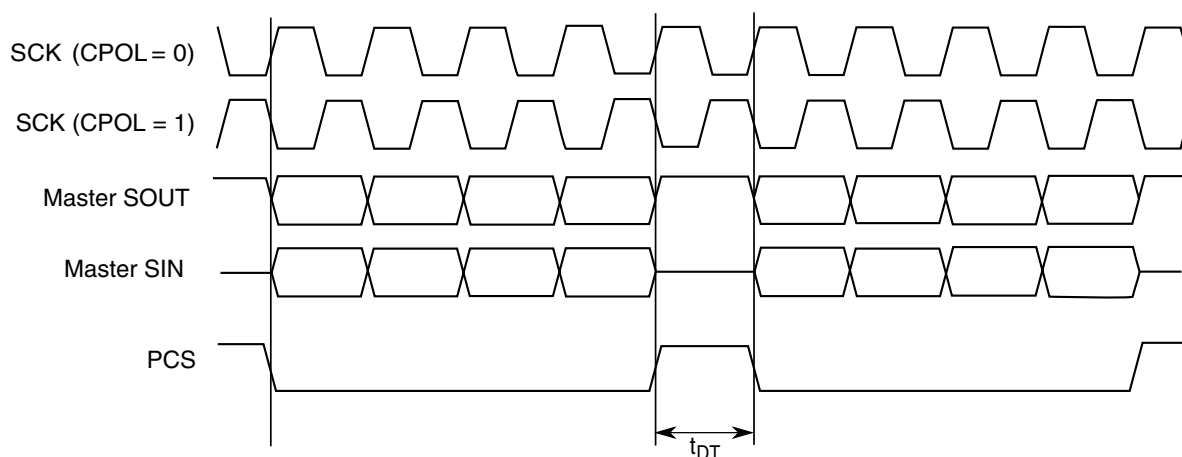


Figure 42-61. Continuous SCK Timing Diagram (CONT=0)

If the CONT bit in the TX FIFO entry is set or the DCONT in DSICR0 is set, PCS remains asserted between the transfers. Under certain conditions, SCK can continue with PCS asserted, but with no data being shifted out of SOUT, that is, SOUT pulled high. This can cause the slave to receive incorrect data. Those conditions include:

- Continuous SCK with CONT bit set, but no data in the TX FIFO.
- Continuous SCK with CONT bit set and entering Stopped state (refer to [Start and Stop of module transfers](#)).
- Continuous SCK with CONT bit set and entering Stop mode or Module Disable mode.

The following figure shows timing diagram for Continuous SCK format with Continuous Selection enabled.

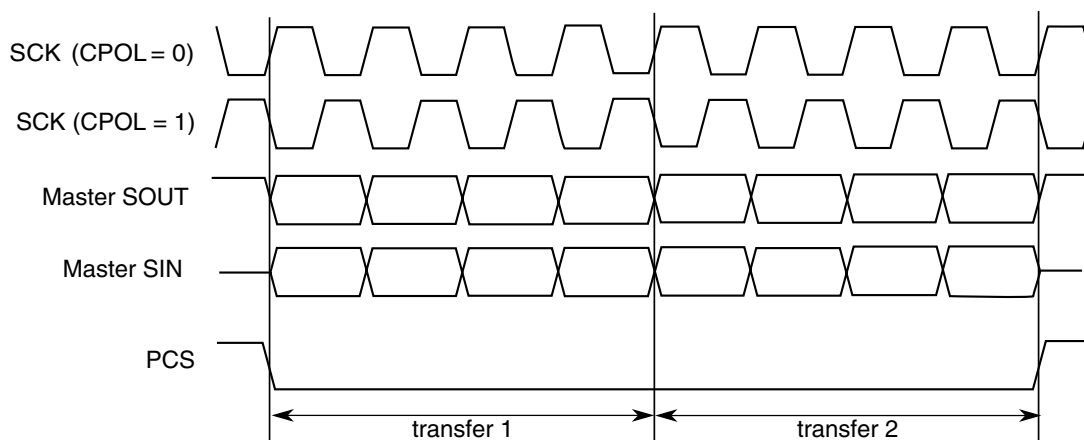


Figure 42-62. Continuous SCK timing diagram (CONT=1)

42.5.8 Slave Mode Operation Constraints

Slave mode logic shift register is buffered. This allows data streaming operation, when the module is permanently selected and data is shifted in with a constant rate.

The transmit data is transferred at second SCK clock edge of the each frame to the shift register if the $\overline{SS}$ signal is asserted and any time when transmit data is ready and $\overline{SS}$ signal is negated.

Received data is transferred to the receive buffer at last SCK edge of each frame, defined by frame size programmed to the CTAR0/1 register. Then the data from the buffer is transferred to the RXFIFO or DDR register.

If the $\overline{SS}$ negates before that last SCK edge, the data from shift register is lost.

42.5.9 Timed Serial Bus (TSB)

The DSPI can be programmed in Timed Serial Bus configuration by setting the TSBC bit in the DSPI_DSICR0 register. See DSPI DSI Configuration Register 0 (DSPI_DSICR0) for details.

TSB configuration provides the Micro Second Channel (MSC) downstream channel support.

The MSC upstream channel is not supported by the DSPI, but can be supported by any available Serial Communication Controller (SCI or UART) in the SoC.

The DSI Continuous Frame Selection Format is not supported by the TSB Mode (DSICR0[DCONT] bit cannot not be asserted).

To work in TSB mode the DSPI must be in master mode and in DSI (DCONF = 0b01) or CSI (DCONF = 0b10) configuration. Both Continuous and Non Continuous Serial Communication Clock (controlled by the DSPI_MCR[CONT_SCKE] bit) are supported in the TSB mode.

The following figure shows the signals used in the TSB interface.

In the TSB configuration the DSPI is able to send from 4 to 34 bits MSC data frames (4 to 32 serialized data bits and up to 2 Data Selection zero bits). The serialized data bits source can be either:

- the DSPI DSI Alternate Serialization Data Register (DSPI_ASDR0), written by the host software,
- Parallel Input pin states latched into the DSPI DSI Serialization Data Register (DSPI_SDR0).

DSPI_DSICR0 TXSS bit or DSPI_SSR0 register bits define the source of the data.

The Least Significant Bits of the DSPI_ASDR0 or DSPI_SDR0 registers are selected to be serialized if the data frame is set to less than 32 bits.

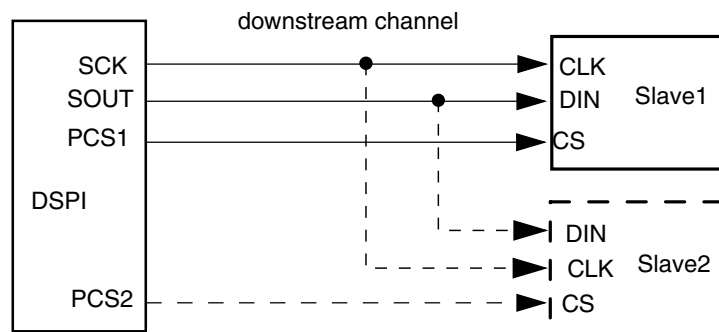


Figure 42-63. DSPI usage in the TSB Configuration

The PCS signals are driven together with SOUT. The t_{CSC} and t_{ASC} delays are not available. Delay after Transfer (DT) is set in SCK clock periods as a binary number formed by concatenation of the DSPI_CTARn PDT and DT fields plus one, allowing to set DT from 1 to 64 serial clock periods. DT field provides least significant bits and PDT field provides most significant bits of the Delay after Transfer.

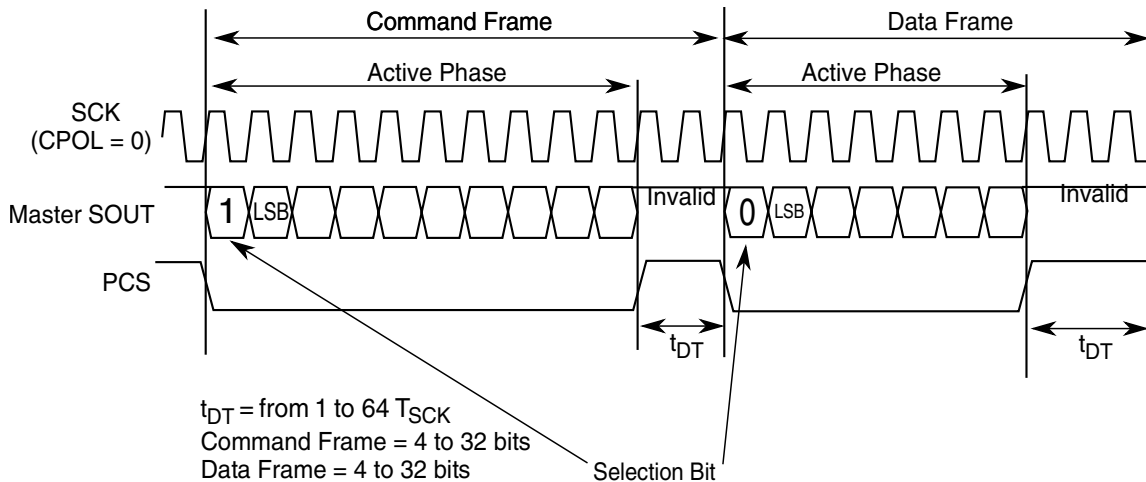


Figure 42-64. TSB Downstream Frames

Above figure shows the two types of MSC downstream frames - command frame, and data frame.

The first transmitted bit, called the selection bit, determines the frame type:

- The selection bit "0" indicates a data frame
- The selection bit "1" indicates a command frame

Data frame may contain up to 2 selection bits to support two external slave devices, (so called dual receiver configuration) or no selection bits at all.

The command frame can be written by software, through SPI TX FIFO, using one or two FIFO entries with help of the CONT bit. The data frame consists of up to 32 bits from the SDR0 or ASDR0 registers and up to two selection bits (0). Number of data bits in the data frame is defined by the DSCICR1[TSBCNT] field.

The selection bit of the MSC command frames (1) can be implemented by software.

The selection bits in the data frames are enabled by DSPI_DSICR1 DSE0 and DSE1 bits. Each DSEn bit set increases the data frame size by one bit.

To comply with MSC specification, set DSPI_CTARn[LSBFE] to transmit least significant bit first.

Regardless the LSBFE bit setting, the Data Frame Selection Bits, if enabled, are always transmitted first, before corresponded data sub frames.

42.5.9.1 MSC Dual Receiver Support with PCS Switch Over

When in TSB mode it is possible to switch the set of PCS signals that are driven during the first part of the frame to a different set of PCS signals during the second part of the frame. The bit, at which this switch over occurs, is defined by FMSZ field of the DSPI_CTARn register, selected by DSICTAS field of the DSICR0 register.

Number of the bits, not including the Data Selection Bit, in the first part of the frame is equal to value of the FMSZ field plus one. During this part of the frame the PCS signal levels are controlled by DSPI_DSICR0 DPCS_n bits, after that by DSPI_DSICR1 DPCS_{1_n} bits.

The PCS switch over occurs at driving edge of the SCK clock output.

The second Data Selection Bit is inserted after the PCS switch over if enabled.

Data Frame with PCS switch over is shown in the following figure.

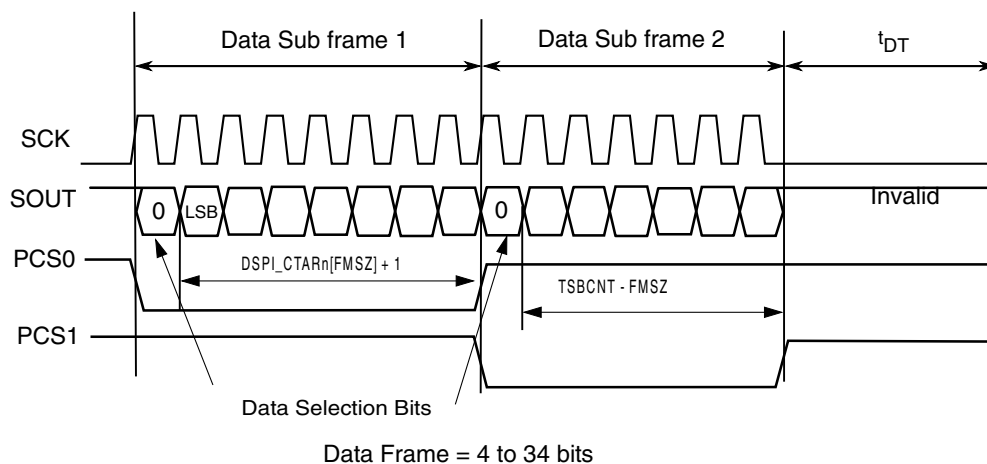


Figure 42-65. TSB Data Frame Format for MSC Dual Receiver Operation

42.5.10 Parity Generation and Check

The module can generate and check parity in the serial frame. The parity bit replaces the last transmitted bit in the frame. The parity is calculated for all transmitted data bits in frame, not including the last data bit that would be transmitted. The parity generation/control is done on frame basis. The registers field setting frame size defines the total

number of bits in the frame, including the parity bit. Thus, to transmit/receive the same number of data bits with parity check, increase the frame size by one versus the same data size frame without the parity check.

Parity can be selected as odd or even. Parity Errors in the received frame set Parity Error flags in the Status register. The Parity Error Interrupt Requests are generated if enabled. The module can be programmed to stop frame transmission in case of a frame reception with parity error.

42.5.10.1 Parity for SPI Frames

When the module is in the master mode the parity generation is controlled by PE and PP bits of the CMD FIFO entries (PUSHR). Setting the PE bit enables parity generation for transmitted SPI frames and parity check for received frames. PP bit defines polarity of the parity bit.

When continuous PCS selection is used to transmit SPI data, two parity generation scenarios are available:

- Generate/check parity for the whole frame
- Generate/check parity for each sub-frame separately.

To generate/check parity for the whole frame set PE bit only in the last command/TX FIFO entry, forming this frame (with the PUSHR register).

To generate/check parity for each sub-frame set PE bit in each command/TX FIFO entry, forming this frame.

If the parity error occurs for received SPI frame, the SR[SPEF] bit is set. If MCR[PES] bit is set, the module stops SPI frames transmission. To resume SPI operation clear the SR[SPEF] or the MCR[PES] bits.

In slave mode the parity is controlled by the PE and PP bits of the CTAR0 register similar to the master mode parity generation without continuous PCS selection.

42.5.10.2 Parity for DSI Frames

When DSPI is in Master Mode, parity generation is controlled by PE and PP bits of the DSPI_DSICR0 register similar to the SPI frames. The parity is calculated and checked for each DSI frame. DSPI_DSICR0[DCONT] bit has no effect on parity generation. In slave mode the parity is controlled by the PE and PP bits of the CTAR1 register.

If the parity error occurs for received DSI frame, the DSPI_SR[DPEF] bit is set. If DSPI_DSICR0[PES] bit is set, the DSPI stops DSI frames transmission. To resume DSI operation clear the DSPI_SR[DPEF] or the DSPI_DSICR0[PES] bits.

42.5.11 Interrupts/DMA requests

The module has several conditions that can generate only interrupt requests and two conditions that can generate interrupt or DMA requests. The following table lists these conditions.

Table 42-82. Interrupt and DMA request conditions

Condition	Flag	Interrupt	DMA
End of Queue (EOQ)	EOQF	Yes	-
TX FIFO Fill	TFFF	Yes	Yes
Transfer Complete	TCF	Yes	-
TX FIFO Underflow	TFUF	Yes	-
RX FIFO Drain	RFDF	Yes	Yes
RX FIFO Overflow	RFOF	Yes	-
SPI Parity Error	SPEF	Yes	-
DSI Parity Error	DPEF	Yes	-
DSI Deserialized Data Match	DDIF	Yes	-

Each condition has a flag bit in the module Status Register (SR) and a Request Enable bit in the DMA/Interrupt Request Select and Enable Register (RSER). Certain flags (as shown in above table) generate interrupt requests or DMA requests depending on configuration of RSER register.

42.5.11.1 End Of Queue interrupt request

The End Of Queue (EOQ) interrupt request indicates that the end of a transmit queue is reached. The module generates the interrupt request when EOQ interrupt requests are enabled (RSER[EOQF_RE]) and the EOQ bit in the executing SPI command is 1.

The module generates the interrupt request when the last bit of the SPI frame with EOQ bit set is transmitted.

42.5.11.2 Transmit FIFO Fill Interrupt or DMA Request

The Transmit FIFO Fill Request indicates that the TX FIFO is not full. The Transmit FIFO Fill Request is generated when the number of entries in the TX FIFO is less than the maximum number of possible entries, and the TFFF_RE bit in the RSER is set. The TFFF_DIRS bit in the RSER selects whether a DMA request or an interrupt request is generated.

NOTE

TFFF flag clears automatically when DMA is used to fill TX FIFO. Configure the DMA to fill only one FIFO location per transfer.

To clear TFFF when not using DMA, follow these steps for every PUSH performed using CPU to fill TX FIFO:

1. Wait until TFFF = 1.
2. Write data to PUSHR using CPU.
3. Clear TFFF by writing a 1 to its location. If TX FIFO is not full, this flag will not clear.

42.5.11.3 Transfer Complete Interrupt Request

The Transfer Complete Request indicates the end of the transfer of a serial frame. The Transfer Complete Request is generated at the end of each frame transfer when the TCF_RE bit is set in the RSER.

42.5.11.4 Transmit FIFO Underflow Interrupt Request

The Transmit FIFO Underflow Request indicates that an underflow condition in the TX FIFO has occurred. The transmit underflow condition is detected only for the module operating in Slave mode and SPI configuration. The TFUF bit is set when the TX FIFO of the module is empty, and a transfer is initiated from an external SPI master. If the TFUF bit is set while the TFUF_RE bit in the RSER is set, an interrupt request is generated.

42.5.11.5 Receive FIFO Drain Interrupt or DMA Request

The Receive FIFO Drain Request indicates that the RX FIFO is not empty. The Receive FIFO Drain Request is generated when the number of entries in the RX FIFO is not zero, and the RFDF_RE bit in the RSER is set. The RFDF_DIRS bit in the RSER selects whether a DMA request or an interrupt request is generated. Configure the DMA to drain only one FIFO location per transfer.

42.5.11.6 Receive FIFO Overflow Interrupt Request

The Receive FIFO Overflow Request indicates that an overflow condition in the RX FIFO has occurred. A Receive FIFO Overflow request is generated when RX FIFO and shift register are full and a transfer is initiated. The RFOF_RE bit in the RSER must be set for the interrupt request to be generated.

Depending on the state of the ROOE bit in the MCR, the data from the transfer that generated the overflow is either ignored or shifted in to the shift register. If the ROOE bit is set, the incoming data is shifted in to the shift register. If the ROOE bit is cleared, the incoming data is ignored.

42.5.11.7 SPI Frame Parity Error Interrupt Request

The SPI Frame Parity Error Flag indicates that a SPI frame with parity error had been received. The SPEF_RE bit in the RSER must be set for the interrupt request to be generated.

42.5.11.8 DSI Frame Parity Error Interrupt Request

The DSI Frame Parity Error Flag indicates that a DSI frame with parity error has been received. The DPEF_RE bit in the DSPI_RSER must be set for the interrupt request to be generated.

42.5.11.9 Deserialized Data Match Interrupt Request

The Deserialized Data Match Flag (DDIF) indicates that a DSI frame matches DSPI_DPIR0 data, masked with DSPI_DIMR0, had been received. The DDIF_RE bit in the DSPI_RSER must be set for the request to be generated.

42.5.12 Power saving features

The module supports following power-saving strategies:

- External Stop mode
- Module Disable mode – Clock gating of non-memory mapped logic

42.5.12.1 Stop mode (External Stop mode)

This module supports the Stop mode protocol. When a request is made to enter External Stop mode, the module acknowledges the request. If a serial transfer is in progress, then this module waits until it reaches the frame boundary before it is ready to have its clocks shut off. While the clocks are shut off, this module's memory-mapped logic is not accessible. This also puts the module in STOPPED state. The SR[TXRXS] bit is cleared to indicate STOPPED state. The states of the interrupt and DMA request signals cannot be changed while in External Stop mode.

42.5.12.2 Module Disable mode

Module Disable mode is a block-specific mode that the module can enter to save power. Host CPU can initiate the Module Disable mode by setting the MDIS bit in the MCR. The Module Disable mode can also be initiated by hardware.

When the MDIS bit is set, the module negates the Clock Enable signal at the next frame boundary. Once the Clock Enable signal is negated, it is said to have entered Module Disable Mode. This also puts the module in STOPPED state. The SR[TXRXS] bit is cleared to indicate STOPPED state. If implemented, the Clock Enable signal can stop the clock to the non-memory mapped logic. When Clock Enable is negated, the module is in a dormant state, but the memory mapped registers are still accessible. Certain read or write operations have a different effect when the module is in the Module Disable mode. Reading the RX FIFO Pop Register does not change the state of the RX FIFO. Similarly, writing to the PUSH Register does not change the state of the TX FIFO or CMD FIFO. Clearing either of the FIFOs has no effect in the Module Disable mode. Changes to the DIS_TXF and DIS_RXF fields of the MCR have no effect in the Module Disable mode. In the Module Disable mode, all status bits and register flags in the module return the correct values when read, but writing to them has no effect. Writing to the TCR during Module Disable mode has no effect. Interrupt and DMA request signals cannot be cleared while in the Module Disable mode.

42.6 Initialization/application information

This section describes how to initialize the module.

42.6.1 How to manage queues

The queues are not part of the module, but it includes features in support of queue management. Queues are primarily supported in SPI configuration.

1. When module executes last command word from a queue, the EOQ bit in the command word is set to indicate it that this is the last entry in the queue.
2. At the end of the transfer, corresponding to the command word with EOQ set is sampled, the EOQ flag (EOQF) in the SR is set.
3. The setting of the EOQF flag disables serial transmission and reception of data, putting the module in the Stopped state. The TXRXS bit is cleared to indicate the Stopped state.
4. The DMA can continue to fill TX FIFO until it is full or step 5 occurs.
5. Disable DMA transfers by disabling the DMA enable request for the DMA channel assigned to TX FIFO (and CMD FIFO) and RX FIFO. This is done by clearing the corresponding DMA enable request bits in the DMA Controller.
6. Ensure all received data in RX FIFO has been transferred to memory receive queue by reading the RXCNT in SR or by checking RFDF in the SR after each read operation of the POPR.
7. Modify DMA descriptor of TX and RX channels for new queues
8. Flush TX FIFO (and CMD FIFO) by writing a 1 to the CLR_TXF bit in the MCR. Flush RX FIFO by writing a '1' to the CLR_RXF bit in the MCR.
9. Clear transfer count either by setting CTCNT bit in the command word of the first entry in the new queue or via CPU writing directly to SPI_TCNT field in the TCR.
10. Enable DMA channel by enabling the DMA enable request for the DMA channel assigned to the module TX FIFO, (and CMD FIFO) and RX FIFO by setting the corresponding DMA set enable request bit.
11. Enable serial transmission and serial reception of data by clearing the EOQF bit.

42.6.2 Switching Master and Slave mode

When changing modes in the module, follow the steps below to guarantee proper operation.

1. Halt it by setting MCR[HALT].
2. Clear the transmit and receive FIFOs by writing a 1 to the CLR_TXF and CLR_RXF bits in MCR.
3. Set the appropriate mode in MCR[MSTR] and enable it by clearing MCR[HALT].

42.6.3 Initializing Module in Master/Slave Modes

Once the appropriate mode in MCR[MSTR] is configured, the module is enabled by clearing MCR[HALT]. It should be ensured that module Slave is enabled before enabling it's Master. This ensures the Slave is ready to be communicated with, before Master initializes communication.

42.6.4 Baud rate settings

The following table shows the baud rate that is generated based on the combination of the baud rate prescaler PBR and the baud rate scaler BR in the CTARs. The values calculated assume a 100 MHz protocol frequency and the double baud rate DBR bit is cleared.

Table 42-83. Baud rate values (bps)

		Baud rate divider prescaler values			
		2	3	5	7
Baud Rate Scaler Values	2	25.0M	16.7M	10.0M	7.14M
	4	12.5M	8.33M	5.00M	3.57M
	6	8.33M	5.56M	3.33M	2.38M
	8	6.25M	4.17M	2.50M	1.79M
	16	3.12M	2.08M	1.25M	893k
	32	1.56M	1.04M	625k	446k
	64	781k	521k	312k	223k
	128	391k	260k	156k	112k
	256	195k	130k	78.1k	55.8k
	512	97.7k	65.1k	39.1k	27.9k
	1024	48.8k	32.6k	19.5k	14.0k
	2048	24.4k	16.3k	9.77k	6.98k
	4096	12.2k	8.14k	4.88k	3.49k
	8192	6.10k	4.07k	2.44k	1.74k
	16384	3.05k	2.04k	1.22k	872
	32768	1.53k	1.02k	610	436

42.6.5 Delay settings

The following table shows the values for the Delay after Transfer (t_{DT}) and CS to SCK Delay (T_{CSC}) that can be generated based on the prescaler values and the scaler values set in the CTARs. The values calculated assume a 100 MHz protocol frequency.

NOTE

The clock frequency mentioned above is given as an example in this chapter. See the clocking chapter for the frequency used to drive this module in the device.

This table does not apply for TSB Continuous mode.

Table 42-84. Delay values

		Delay prescaler values			
		1	3	5	7
Delay scaler values	2	20.0 ns	60.0 ns	100.0 ns	140.0 ns
	4	40.0 ns	120.0 ns	200.0 ns	280.0 ns
	8	80.0 ns	240.0 ns	400.0 ns	560.0 ns
	16	160.0 ns	480.0 ns	800.0 ns	1.1 μ s
	32	320.0 ns	960.0 ns	1.6 μ s	2.2 μ s
	64	640.0 ns	1.9 μ s	3.2 μ s	4.5 μ s
	128	1.3 μ s	3.8 μ s	6.4 μ s	9.0 μ s
	256	2.6 μ s	7.7 μ s	12.8 μ s	17.9 μ s
	512	5.1 μ s	15.4 μ s	25.6 μ s	35.8 μ s
	1024	10.2 μ s	30.7 μ s	51.2 μ s	71.7 μ s
	2048	20.5 μ s	61.4 μ s	102.4 μ s	143.4 μ s
	4096	41.0 μ s	122.9 μ s	204.8 μ s	286.7 μ s
	8192	81.9 μ s	245.8 μ s	409.6 μ s	573.4 μ s
	16384	163.8 μ s	491.5 μ s	819.2 μ s	1.1 ms
	32768	327.7 μ s	983.0 μ s	1.6 ms	2.3 ms
	65536	655.4 μ s	2.0 ms	3.3 ms	4.6 ms

42.6.6 Calculation of FIFO pointer addresses

Complete visibility of the FIFO contents is available through the FIFO registers, and valid entries can be identified through a memory-mapped pointer and counter for each FIFO. The pointer to the first-in entry in each FIFO is memory mapped. For the TX FIFO the first-in pointer is the Transmit Next Pointer (TXNXTPTR). For the CMD FIFO the first-in pointer is the Command Next Pointer (CMDNXTPTR). For the RX FIFO the first-in pointer is the Pop Next Pointer (POPNXTPTR). The following figure illustrates the concept of first-in and last-in FIFO entries along with the FIFO Counter. The TX FIFO is chosen for the illustration, but the concepts carry over. See [Transmit First In First Out \(TX FIFO\) buffering mechanism](#), [Command First In First Out \(CMD FIFO\) Buffering Mechanism](#) and [Receive First In First Out \(RX FIFO\) buffering mechanism](#) for details on the FIFO operation.

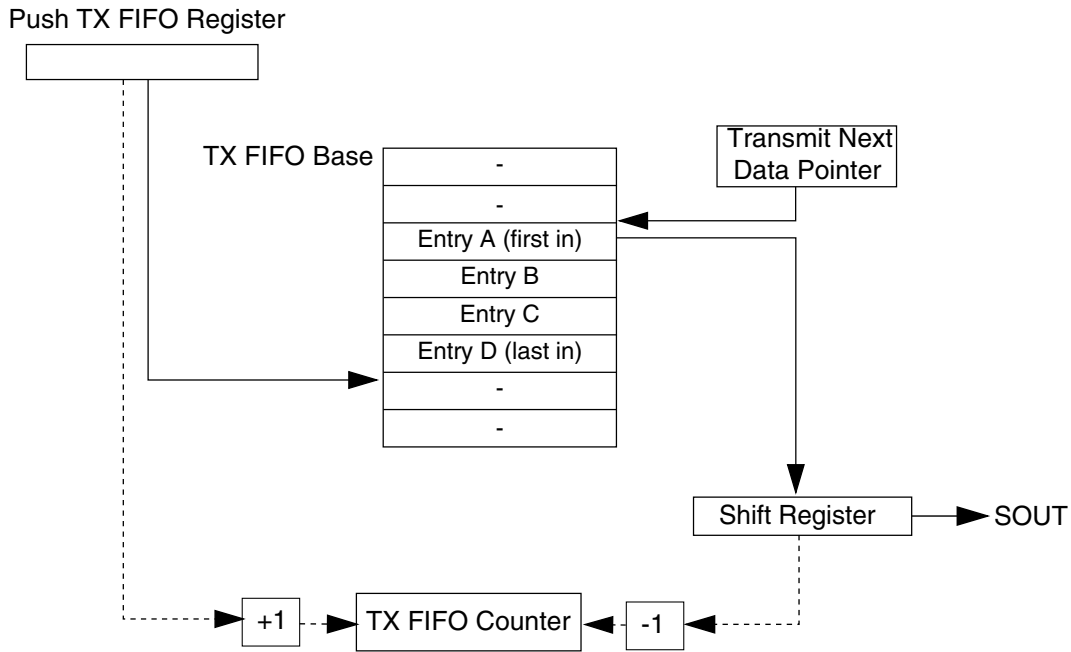


Figure 42-66. TX FIFO pointers and counter

42.6.6.1 Address Calculation for the First-in Entry and Last-in Entry in the TX FIFO

The memory address of the first-in entry in the TX FIFO is computed by the following equation:

$$\text{First-in EntryAddress} = \text{TXFIFOBase} + (4 \times \text{TXNXTPTR})$$

The memory address of the last-in entry in the TX FIFO is computed by the following equation:

$$\text{Last-inEntryaddress} = \text{TXFIFOBase} + 4 \times (\text{TXCTR} + \text{TXNXTPTR} - 1) \bmod (\text{TXFIFOdepth})$$

TX FIFO Base - Base address of TX FIFO

TXCTR - TX FIFO Counter

TXNXTPTR - Transmit Next Pointer

TX FIFO Depth - Transmit FIFO depth, implementation specific

42.6.6.2 Address Calculation for the First-in Entry and Last-in Entry in the CMD FIFO

The memory address of the first-in entry in the CMD FIFO is computed by the following equation:

$$\text{First-in EntryAddress} = \text{TXFIFOBase} + (4 \times \text{TXNXTPTR})$$

The memory address of the last-in entry in the CMD FIFO is computed by the following equation:

$$\text{Last-inEntryaddress} = \text{TXFIFOBase} + 4 \times (\text{TXCTR} + \text{TXNXTPTR} - 1) \bmod (\text{TXFIFOdepth})$$

CMD FIFO Base - Base address of CMD FIFO

CMDCTR - CMD FIFO Counter

CMDNXTPTR - Command Next Pointer

CMD FIFO Depth - Command FIFO depth, implementation specific

42.6.6.3 Address Calculation for the First-in Entry and Last-in Entry in the RX FIFO

The memory address of the first-in entry in the RX FIFO is computed by the following equation:

$$\text{First-in EntryAddress} = \text{RX FIFOBase} + (4 \times \text{POPNXTPTR})$$

The memory address of the last-in entry in the RX FIFO is computed by the following equation:

$$\text{Last-inEntryaddress} = \text{RX FIFO Base} + 4 \times (\text{RXCTR} + \text{POPNXTPTR} - 1) \bmod (\text{RXFIFOdepth})$$

RX FIFO Base - Base address of RX FIFO

RXCTR - RX FIFO counter

POPNXTPTR - Pop Next Pointer

RX FIFO Depth - Receive FIFO depth, implementation specific

Chapter 43

Zipwire

43.1 Chip-specific Zipwire information

The following table provide the values for various Zipwire parameters as implemented in this device. Each parameter is given a variable name that is used in various sections of this chapter.

Table 43-1. Chip-specific parameters

Parameter		Variable name	Value
LFAST maximum baud rate		c	See the product Data Sheet
Bit time		d	$1/c$
XBAR frequency		x	PLAT_CLK
SIPI frequency		y	PLAT_CLK
No. of XBAR cycles		n	7
		m	6
SIPI address time	Read	z	$13/y + n/x$
	Write	v	$13/y + m/x$

43.2 Overview

The SIPI and LFAST modules work together as a single unit called Zipwire. The LFAST portion of the two modules allows for high speed inter-device communications. The SIPI allows memory to be shared between devices which have SIPI and LFAST communication modules.

The LFAST can operate in either slave or master mode configurations. The node running in master mode controls the serial link, but SIPI, in both master and slave modes, can act as both initiator and target for SIPI commands simultaneously. Please see the SIPI and LFAST chapters for detailed information on module configuration and functionality.

43.3 Introduction

Zipwire is a group of modules that allow one MCU to have a fast, low pin count, serial communication link directly into the memory mapped peripherals and/or memories of another MCU or smart ASIC. Zipwire is implemented in hardware so there is no CPU load for the initiator or target node. Zipwire supports 8-bit, 16-bit or 32-bit reads and writes to any 32-bit address at the target node.

Zipwire architecture is fully pipelined and support multiple outstanding commands to allow maximum use of the serial link bandwidth.

The serial link runs at a high speed using LVDS physical layer. One LVDS pair for Tx and another pair for Rx, with a separate LFAST reference clock for a total of five pins.

Zipwire also supports a streaming mode for the transfer of large blocks of data.

In streaming mode Zipwire has a high transfer rate.

43.4 Zipwire Block Diagram

[Figure 43-1](#) shows two MCUs with Zipwire connected using a five wire interface. Both MCUs support Target and Initiator modes of Zipwire. The SIPI Bus Master Interface is used for all Target mode transactions. The SIPI Register Interface is used for all Initiator Transactions and initial setup. The LFAST Register Interface is used for Initial Setup.

The diagram shows the LFAST reference clock coming from MCU B clock system. The Ref_Clock can come from either MCU. It is user configurable, but it must be the same clock that drives both LFAST modules.

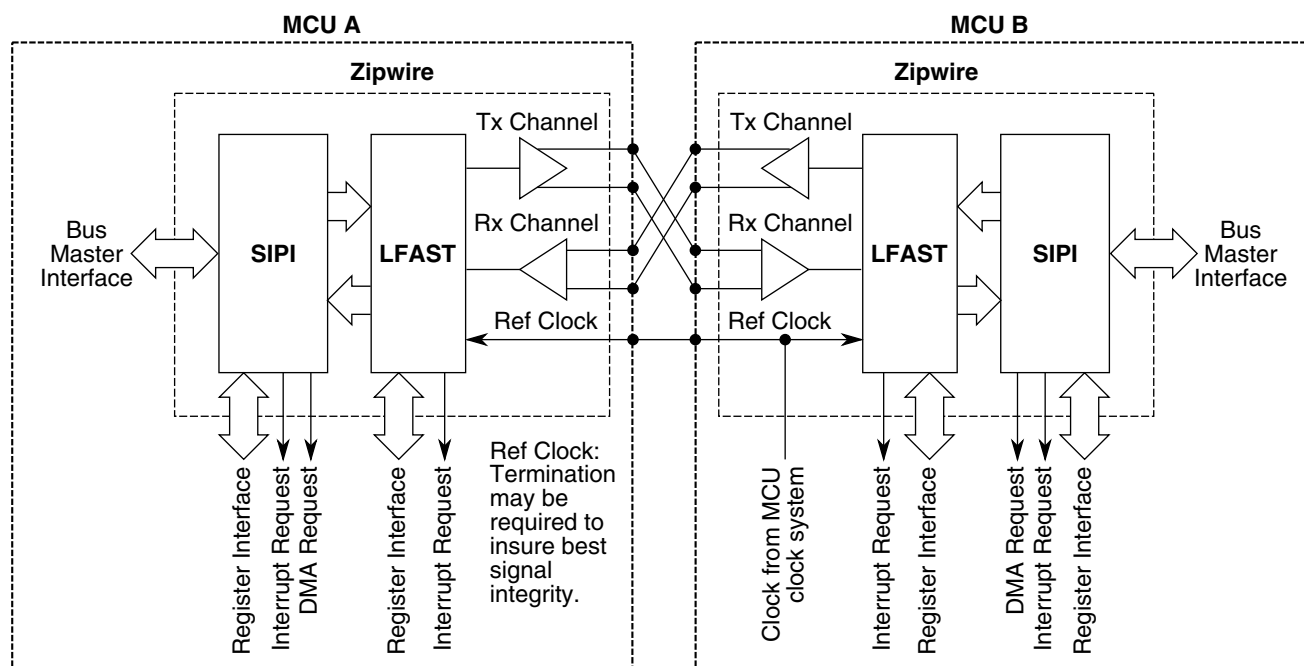


Figure 43-1. Zipwire connection diagram

43.5 Architecture

Zipwire is composed of several modules and system resources. The physical layer is LVDS with common mode voltage and defined swing.

The transport layer is a protocol called LFAST. LFAST is an asynchronous protocol, using non-return to zero encoding. The LFAST protocol is composed of the following:

- A fixed 16-bit sync frame to allow the receiver to detect the optimal point to sample the incoming data.
- Followed by an 8-bit LFAST header, that defines the channel number and the size of the LFAST payload.
- Finally the payload, which can be between 8 and 288 bits.

The application layer protocol is called Serial Inter Processor Interface (SIPI). SIPI runs on top of LFAST and is fully encapsulated within the LFAST payload. SIPI uses four fixed sizes of LFAST payload in Zipwire:

- 32-bit
- 64-bit

- 96-bit
- 288-bit

The SIPI protocol implements a suite of commands initiated by one MCU, for reading and writing any 32-bit address location in another connected MCU. The SIPI protocol allows either or both MCUs to initiate commands. The protocol also supports interrupt requests from one MCU to the other.

The SIPI module implements both the initiator part of the protocol and the recipient part of the protocol. The module implements four SIPI Initiator channels that can run independently of each other and can run simultaneously. The SIPI module is a bus master on the low speed XBAR. As the target MCU for a command from the initiating MCU, SIPI can perform read and write accesses to any address location within the target MCU. The software running on the local MCU, should configure the system MPU to allow/deny memory accesses to MCU memory and resources as required.

The initiator part of the module is connected to the DMA controller and is able to generate DMA requests as a command is completed. This allows a series of read or write commands to be queued in the Initiator MCU and executed by DMA and SIPI without CPU intervention at either Initiator or Target MCU.

43.6 Zipwire interconnections

LFAST has the following connections:

- Tx and Rx configuration is controlled by LFAST Control registers.
- Peripheral Bridge interface (PBRIDGE) — Allows software to read and write configuration registers.
- Tx Data Port/Rx Data Port — Directly connected to the SIPI module. Allows received data to be efficiently transferred to SIPI and transmit data to be transferred from SIPI.
- Interrupt Request connections — Allows the module to flag to the CPU when it requires servicing.

SIPI has the following connections:

- Peripheral Bridge interface (PBRIDGE) — Allows software to read and write configuration registers and SIPI Interface Registers.

- DMA connections — Allows SIPI command sequences to be queues and initiated without CPU intervention.
- Crossbar master port — Allows SIPI to execute requested commands to read and write MCU address space.
- Tx Data Port/Rx Data Port — Directly connected to the LFAST module. Allows received data to be efficiently transferred from LFAST and transmit data to be transferred to LFAST.
- Interrupt Request connections — Allows the module to flag to the CPU when it requires servicing.

43.7 Zipwire performance

Two aspects of performance are considered:

- Bandwidth — The rate at which data can be read or written between two nodes. It assumes that read or write commands from the Initiator node, can be sent continuously at the highest speed the Target node can consume those commands.
- Latency — The time from the Initiator node sending a read or write command to the Initiator node receiving back the read data or write acknowledge.

43.7.1 Read performance

A Zipwire read operation consists of three stages:

- Initiator sends SIPI Read command to Target.
- Target parses the received command and runs a master bus cycle to read the data.
- Target sends the SIPI Read response back to the Initiator.

A SIPI Read command consists of:

- 16-bit header
- 32-bit read address
- 16-bit CRC
- 64 bits total

A SIPI Read response consists of:

- 16-bit header
- 32-bit read data
- 16-bit CRC
- 64 bits total

The SIPI message is encapsulated in a LFAST frame where the LFAST payload is the size of the SIPI message.

The LFAST frame consists of:

- 16-bit synchronisation header
- 8-bit header
- 64-bit payload
- 1-bit stop bit
- 89 bits total

LFAST Baud rate is c Mbaud which yields a bit time of d ns

Therefore, an 89-bit LFAST transmission of a SIPI 64-bit Read command or 64-bit Read response takes $89d$ ns.

The SIPI module takes thirteen system clock cycles to parse a command and create a Read response, plus the time to run the master bus cycle. The time to run the master bus cycle will vary depending on the memory being read.

Assumptions:

- Average of $n \times$ XBAR cycles to read different memories.
- XBAR clocked at x MHz
- SIPI clocked at y MHz

Therefore, time for SIPI to read an address location is:

$$(13 \times y \text{ MHz clocks}) + (n \times x \text{ MHz clocks}) = z \text{ ns}$$

43.7.1.1 Read bandwidth

The Zipwire target node has a three message deep receive FIFO and a three message deep transmit FIFO. So, the Target node can simultaneously be receiving a command, processing a command, and sending the Read response.

The time to transmit a command or response is less than the time to process the command. Therefore, the bandwidth is limited by the time to process the message.

$$\text{Read Bandwidth} = 4 \text{ bytes in } z \text{ ns} = 4 \times z \text{ MB/second}$$

43.7.1.2 Read latency

The latency is the time taken to send a Read command, process the command, and send a Read response.

$$\text{Read Latency} = 89d \text{ ns} + z \text{ ns} + 89d \text{ ns} = z + 178d \text{ ns}$$

43.7.2 Write performance

A Zipwire write operation consists of three stages:

- Initiator sends SIPI Write command to Target.
- Target parses the received command and runs a master bus cycle to write the data.
- Target sends the SIPI Write response back to the Initiator.

A SIPI Write command consists of:

- 16-bit header
- 32-bit write address
- 32-bit write data
- 16-bit CRC
- 96 bits total

A SIPI Write acknowledge consists of:

- 16-bit header

- 16-bit CRC
- 32 bits total

The SIPI message is encapsulated in an LFAST frame where the LFAST payload is the size of the SIPI message.

The LFAST frame consists of:

- 16-bit synchronisation header
- 8-bit header
- 96-bit payload (command) or 32-bit (acknowledge)
- 1-bit stop bit
- 121 bits total (command) or 57 bits (acknowledge)

LFAST Baud rate is c Mbaud which yields a bit time of d ns

Therefore, the time for an LFAST transmission is:

- 121-bit LFAST transmission of a SIPI 96-bit Write Command takes $121d$ ns
- 57-bit LFAST transmission of a SIPI 32-bit Write Acknowledge takes $57d$ ns

The SIPI module takes thirteen system clock cycles to parse a command and create a Write response, plus the time to run the master bus cycle. The time to run the master bus cycle will vary depending on the memory being written.

Assumptions:

- Average of $m \times$ XBAR cycles to write different memories.
- XBAR clocked at x MHz
- SIPI clocked at y MHz

Therefore, the time for SIPI to write an address location is:

$$(13 \times y \text{ MHz clocks}) + (m \times x \text{ MHz clocks}) = v \text{ ns}$$

43.7.2.1 Write bandwidth

The Zipwire target node has a three message deep receive FIFO and a three message deep transmit FIFO. So, the Target node can simultaneously be receiving a command, processing a command, and sending the Write Acknowledge response.

The time to transmit the command is longer than the time to process the command, or the time to transmit the Write Acknowledge. Therefore, the bandwidth is limited by the time to transmit the command.

Write Bandwidth = 4 bytes in ν ns = $4 \times \nu$ MB/second

43.7.2.2 Write latency

The latency is the time taken to send a Write command, process the command, and send a Write Acknowledge.

Write Latency = $121d$ ns + ν ns + $57d$ ns = $\nu + 178d$ ns

Chapter 44

Serial Interprocessor Interface (SIPI)

44.1 Chip-specific SIPI information

This chip has one instance of the SIPI module.

44.1.1 SIPI Chip ID

On this chip, the SIPI Chip ID is the same as the [JTAG ID](#).

44.2 Introduction

The Serial Interprocessor Interface (SIPI) is an application layer protocol which runs on top of the LFAST (LVDS Fast Asynchronous Serial Transmission) module. It is used by the local device to access the shared memory of a remote device. SIPI defines point-to-point full duplex communication between two devices, where LFAST works as a physical medium of communication between both the devices.

44.2.1 Scalability

The SIPI protocol is designed to provide a sophisticated, high bandwidth, multimaster, multi-channel memory interface between 2 devices using few interconnecting signals. But the protocol is designed in such a way that a subset of the protocol can be implemented, where die area is more important than features.

Main scalable features:

- Number of concurrent channels:

- Full Implementation = 4
- Minimum Implementation = 1
- Full implementation has a node both as Initiator and Target of commands. Minimum implementation either Initiator or Target.
- Full Implementation includes a block transfer feature, but this feature is optional.

The rest of this section describes a full implementation of SIPI which includes:

- Advanced High-Performance Bus (AHB-Lite) master interface
- Direct Memory Access (DMA) interface
- LFAST Tx/Rx (transmit/receive) interfaces along with Peripheral Bus Interface (IPS)

44.3 Overview

An instance of SIPI can act as initiator, or target or both. SIPI can access the shared memory directly through its AHB master interface or through its DMA interface. DMA interface is used when the node acts as an initiator while the AHB Master interface will be used when the node acts as target. SIPI has four channels, with one channel (Channel 2) having data streaming capability. Payload width for channel 0, 1 and 3 is 32 bits, and for channel 2 data widths can be 32 bits, or 256 bits when streaming. Any of these channels can be used for DMA access or bus interface access depending on the setting of the SIPI_CCR*n*. CRC encoder calculates the CRC on the command frame. Then the SIPI initiator appends the CRC to the end of the frame before transmission.

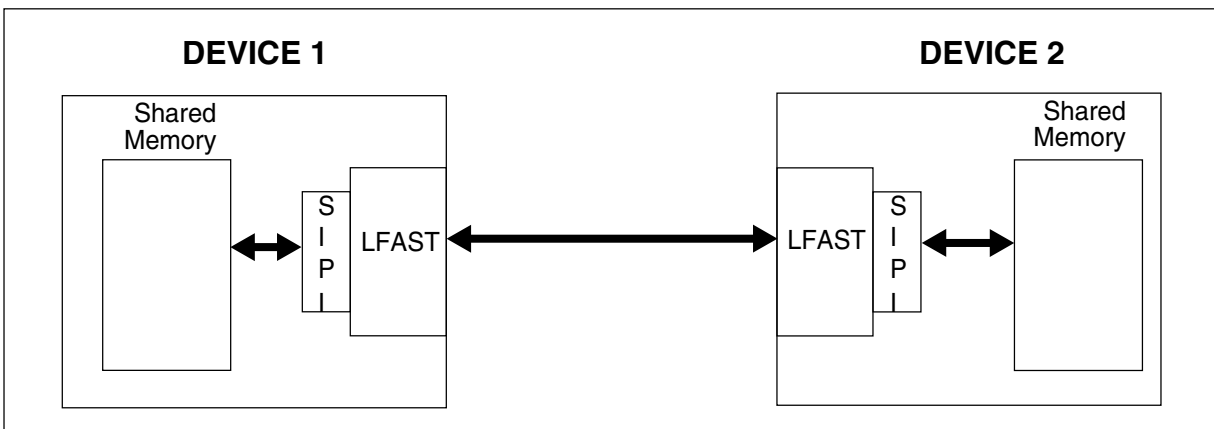
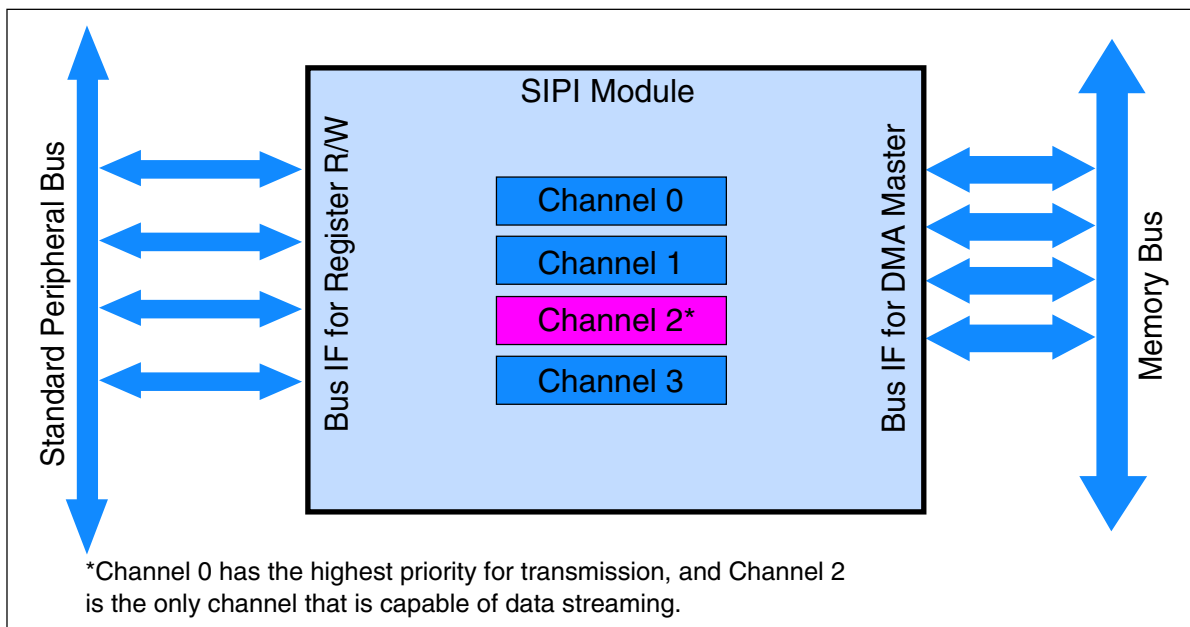
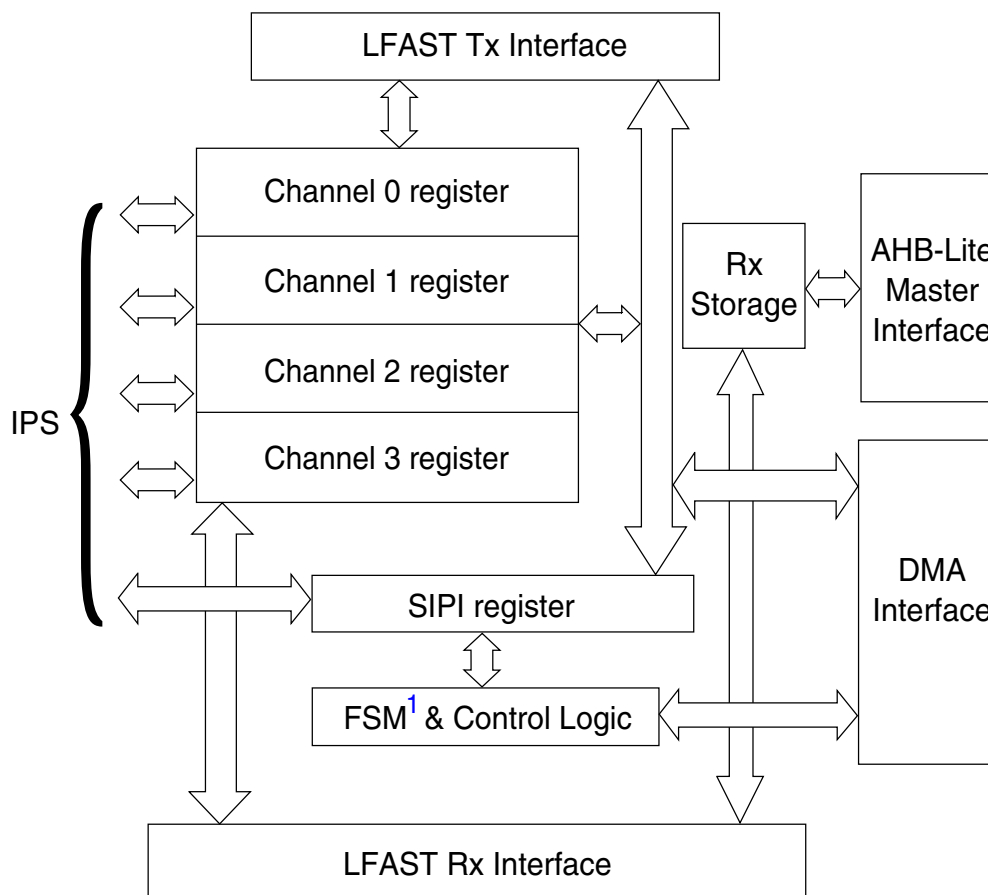


Figure 44-1. Interprocessor communication diagram

**Figure 44-2. SIPI module**

44.4 SIPI block diagram



¹ Finite State Machine

Figure 44-3. SIPI block diagram

44.5 Feature description

This section describes the features of the SIPI module.

44.5.1 Main features

- Point-to-point communication between two devices
- Full duplex communication
- Four channels, including one channel with data streaming capability

- Configurable DMA access for each channel
- CRC protection mechanism
- Timeout protection mechanism
- Fixed priority channel selection
- Data size up to 256 bits on streaming channel
- Common tag pool for assigning sequential transfer IDs to every new transfer
- AHB master interface which is used by target node to access shared memory
- Target node contains a set of nine 32-bit internal registers to store commands
- Up to two outstanding requests are supported at initiator

44.5.2 Standard features

- IPS bus interface (PBRIDGE)
- SPP DMA2x bus interface
- AHB Master Interface
- LFAST Tx/Rx interfaces
- Cyclic Redundancy Check error detection (CRC16)

44.6 SIPI operation from reset

When the SIPI module exits reset, it is operational and target mode is enabled without the need to configure the control registers.

44.7 Functional description

44.7.1 External signals

The SIPI has no chip external signals.

44.7.2 Frame format

All frames have the same general format:

- 16 bit header
- Address, Address and Data, or nothing
- 16-bit CRC

There are 2 main groups of command; read and write. Within those 2 groups are three read/write formats:

- 32-bit
- 16-bit
- 8-bit

Each command generates one of three different responses:

- Read data
- Write acknowledge
- Error

There is one additional command that requests the module ID from the Target node. The ID is a unique 32-bit ID that is specific to a particular device. It is normally the same as the JTAG ID. The sections below illustrate the four different frame formats that are used to implement all the commands and responses.

The number of frames depends on the data buffers (associated with every channel), the frame data is stored in data buffers at the initiator. Address and data are both transmitted in the same order in which they are stored.

44.7.2.1 Register read request

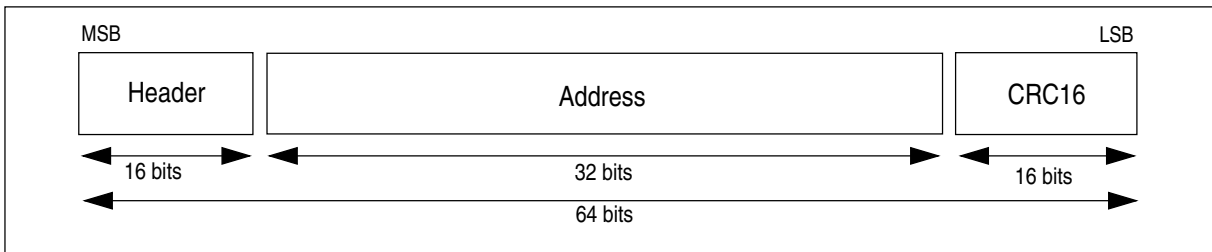


Figure 44-4. SIPI register read request

44.7.2.2 Register read response, ID request response

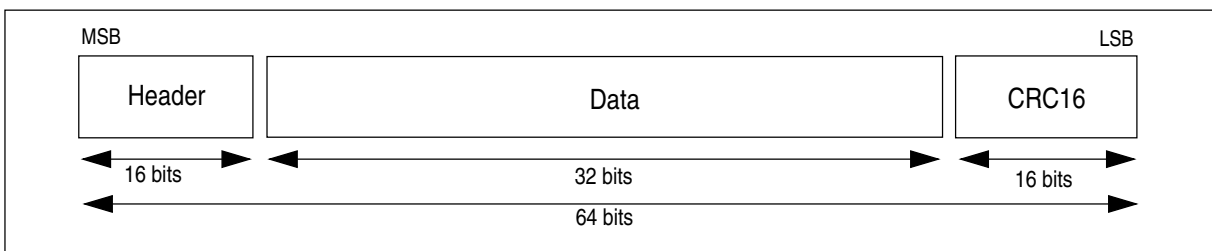


Figure 44-5. SIPI read response

44.7.2.3 Register write request

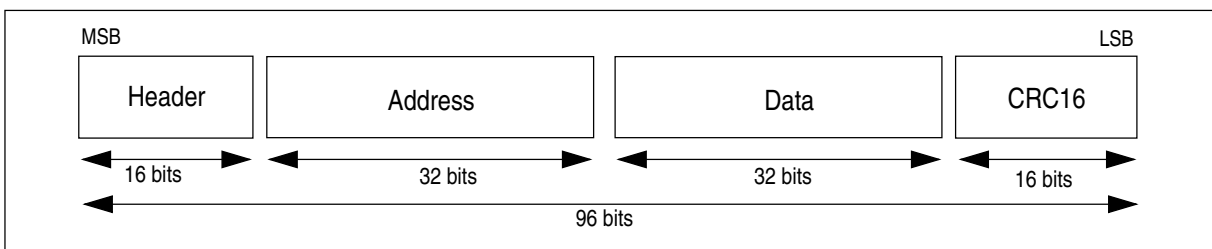


Figure 44-6. SIPI register write request

44.7.2.4 Trigger transfer, ID transfer, write acknowledge and streaming write acknowledge

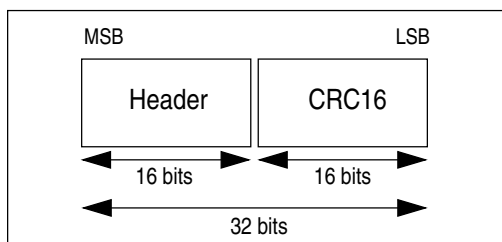


Figure 44-7. SIPI write acknowledge

44.7.2.5 Streaming write request

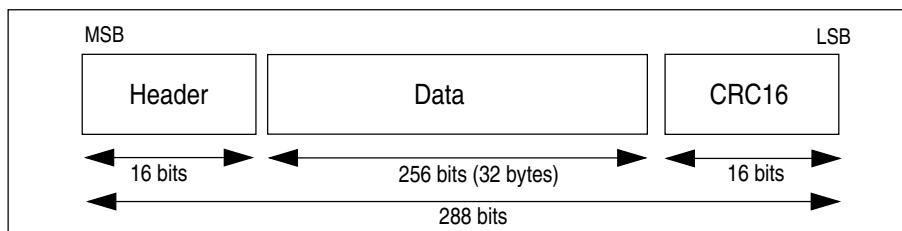


Figure 44-8. Streaming write request format

NOTE

Direct write operations are used to set the streaming address.

Streaming data write is performed using the format shown in [Figure 44-8](#).

44.7.2.6 Header field

Header field contains 16 bits of configuration information. MSB will be transmitted first.

44.7.2.6.1 SIPI header coding

[Figure 44-9](#), [Table 44-1](#), [Figure 44-10](#), and [Table 44-2](#) show how the SIPI header bits are coded.

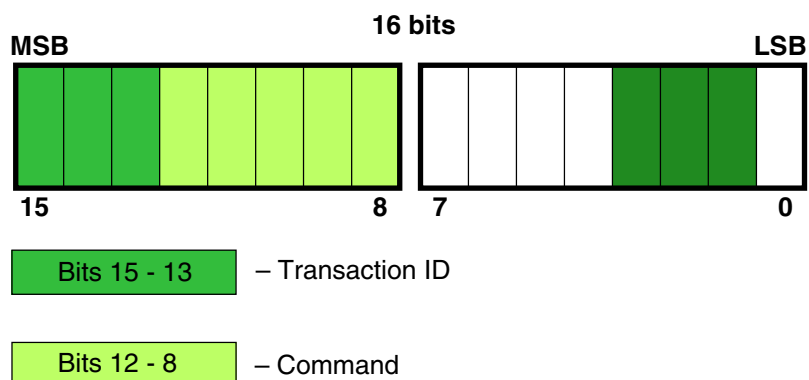
**Figure 44-9. SIPI header coding**

Table 44-1 shows the command coding for the bits[12:8] of the header.

Table 44-1. SIPI header command coding

b[12:8] (hex)	Command	Payload Size
00	Read 8 bits	64
01	Read 16 bits	64
02	Read 32 Bits	64
03	Reserved	—
04	Write 8 bits with ACK	96
05	Write 16 bits with ACK	96
06	Write 32 bits with ACK	96
07	Reserved	—
08	ACK – OK	32
09	ACK – Fault	32
0A	Read Answer – OK	64
0B	Reserved	—
0C	Trigger comm and with ACK	32
0D	Reserved	—
...	...	...
11	Reserved	—
12	ID Register Read Request	32
13	Reserved	—
...	...	...
16	Reserved	—
17	Stream Data with ACK (32 bytes)	288
18	Reserved	—
...	...	...
1F	Reserved	—

Figure 44-10 shows the channel number field in the header and Table 44-2 shows the channel number coding.

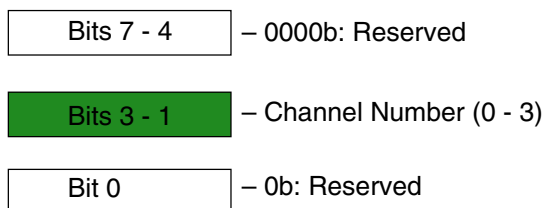


Figure 44-10. SIPI header channel field

Table 44-2. SIPI header channel number coding

SIPI channel name	Channel number coding in header(s)		Comment
	SIPI channel coding select field (SIPI_MCR[CHNSB])		
	Code table I (SIPI_MCR[CHNSB] = 1)	Code table II (SIPI_MCR[CHNSB] = 0)	
0 (Channel A)	000b	100b	In use
1 (Channel B)	001b	101b	In use
2 (Channel C)	010b	110b	In use
3 (Channel D)	011b	111b	In use
4 (Channel E)	100b	000b	Reserved
5 (Channel F)	101b	001b	Reserved
6 (Channel G)	110b	010b	Reserved
7 (Channel H)	111b	011b	Reserved

44.7.2.7 Address field

The address field is 32 bits wide with the MSB transmitted first.

44.7.2.8 Payload field

Table 44-3 shows the payload sizes of LFAST frames.

Table 44-3. Payload size of LFAST channel frame

LFAST Code	SIPI Code	LFAST Payload Size (bits)	LFAST Payload Size (bytes)
000b	—	8	1
001b	—	32	4
010b	010b	64	8
011b	011b	96	12
100b	100b	128	16

Table continues on the next page...

Table 44-3. Payload size of LFAST channel frame (continued)

LFAST Code	SIPI Code	LFAST Payload Size (bits)	LFAST Payload Size (bytes)
101b	101b	256	32
110b	110b	512	64
111b	111b	288	36

Table 44-4 shows the converted coding of LFAST for a given SIPI code.

Table 44-4. Converted coding of LFAST channel code for SIPI headers

LFAST Code	SIPI Code	Channel (hex) ¹
0100b	100b	A
0101b	101b	B
0110b	110b	C
0111b	111b	D
1000b	000b	E (not used by SIPI)
1001b	001b	F (not used by SIPI)
1010b	010b	G (not used by SIPI)
1011b	011b	H (not used by SIPI)

1. SIPI channel 0 sends all commands on LFAST channel A, commands received on LFAST channel A, are processed and the response sent back on LFAST channel A.
 SIPI channel 1 sends all commands on LFAST channel B, commands received on LFAST channel B, are processed and the response sent back on LFAST channel B.
 SIPI channel 2 sends all commands on LFAST channel C, commands received on LFAST channel C, are processed and the response sent back on LFAST channel C.
 SIPI channel 3 sends all commands on LFAST channel D, commands received on LFAST channel D, are processed and the response sent back on LFAST channel D.

44.7.2.9 CRC field

CRC field is 16 bits wide with calculation always enabled using CRC-16-CCITT syndrome ($x^{16} + x^{12} + x^5 + 1$). MSB is sent first in the data stream.

44.7.2.9.1 CRC field examples

44.7.2.9.1.1 Example 1 – 32 bit write on channel 1 with ID1

- Header = 260Ah
- Address = 1122_3344h
- Data = CCDD_EEFFh
- CRC = BF7Dh

44.7.2.9.1.2 Example 2 – 32 bit read on channel 2 with ID2

- Header = 420Ch
- Address = 89AB_CDEFh
- CRC = 6B80h

44.7.2.9.1.3 Example 3 – Event command on channel 3 with ID3

- Header = 6C0Eh
- CRC = B286h

44.8 Transfer types

This section describes the available transfer types of the SIPI module. The SIPI frame is inserted inside the payload of the LFAST frame as shown in the figures of the following examples.

44.8.1 Read transfer

A Read transfer can be of two types:

- Read Request Transfer (at initiator node)
- Read Response Transfer (at target node)

44.8.1.1 Register read request transfer

If there is a read request transfer, the initiator node will send header, address and CRC bits as shown in [Figure 44-11](#).

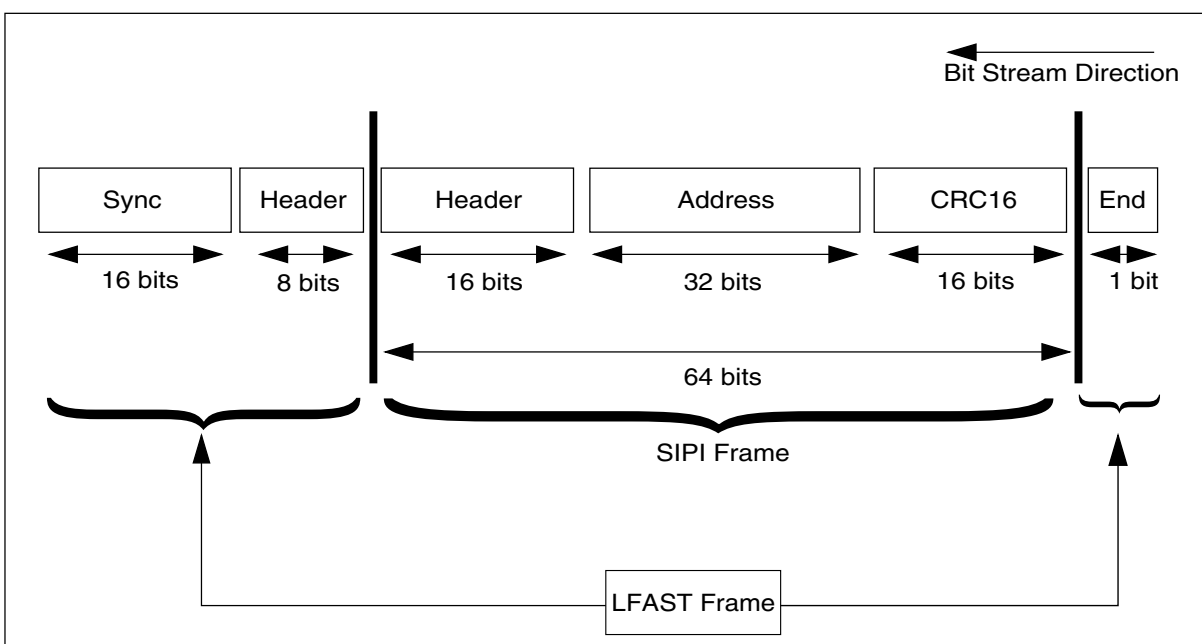


Figure 44-11. Read request transfer

44.8.2 Register read answer transfer

In response to a read request by the initiator node, the target node will send header, payload and CRC16. Data transfer could be in 8-bit, 16-bit or 32-bit modes (see [Figure 44-12](#)). In the case of 8-bit or 16-bit modes, copies of the SIPI data is sent in the payload (see [Figure 44-13](#)).

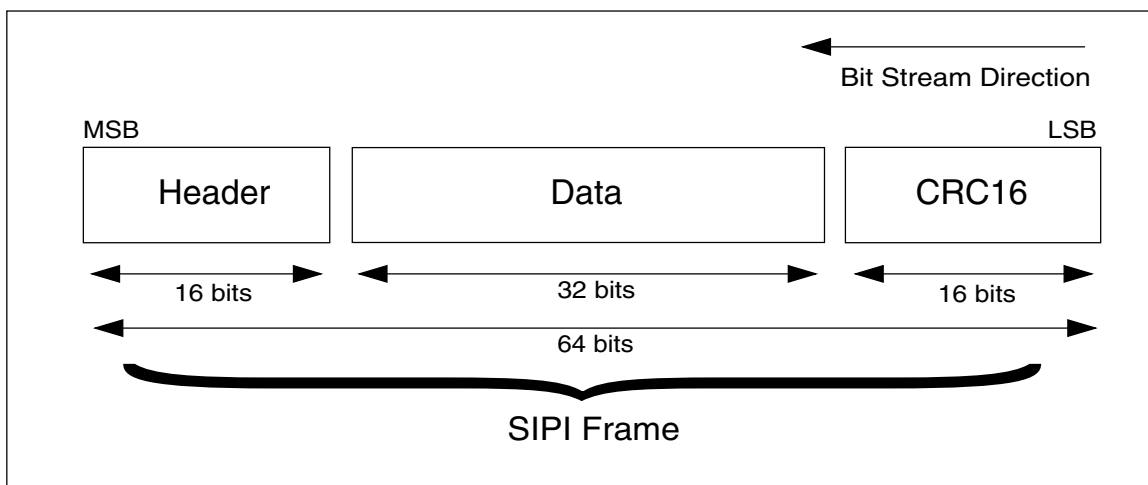


Figure 44-12. Read answer transfer

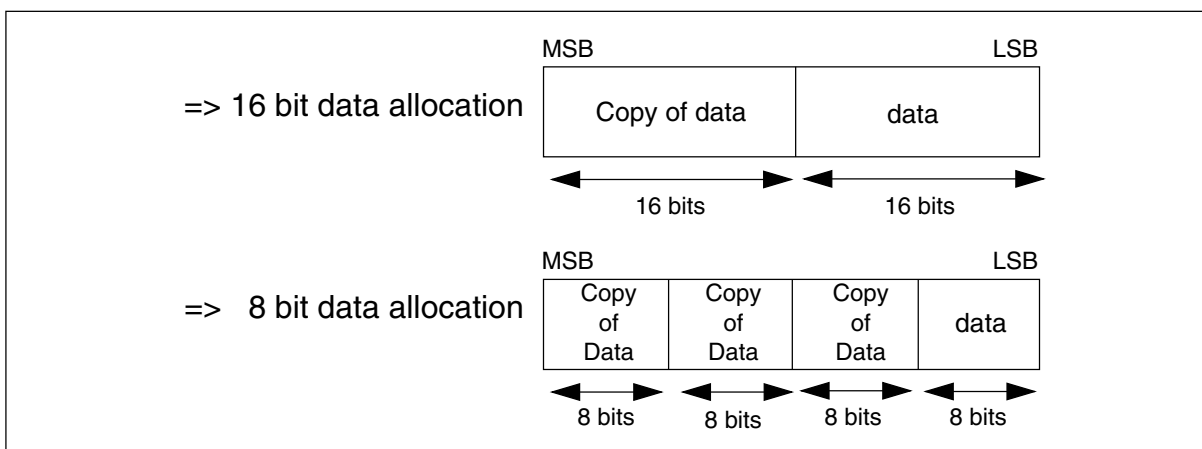


Figure 44-13. Data allocation

Note

- For an 8-bit transfer, address bits [31:2] are used as address and bits [1:0] are used as byte enables.
 - Bits[1:0]:
 - 00 - byte 3 enabled (MSB)
 - 01 - byte 2 enabled
 - 10 - byte 1 enabled
 - 11 - byte 0 enabled (LSB)
- For a 16-bit transfer, address bits [31:1] are used as address and bit [0] is used as halfword enable.
 - Bit[0]:
 - 0 - halfword 1 enabled (MSB)
 - 1 - halfword 0 enabled (LSB)

44.8.3 Register Write transfer

Register Write transfer can be of two types:

- Normal write transfer - channels 0, 1, 2 and 3
- Streaming data transfer - channel 2 only

A Register Write transfer can be done through Normal Write transfer on channels 0, 1, 2 and 3 (see [Normal write transfer](#)) as shown in [Figure 44-14](#).

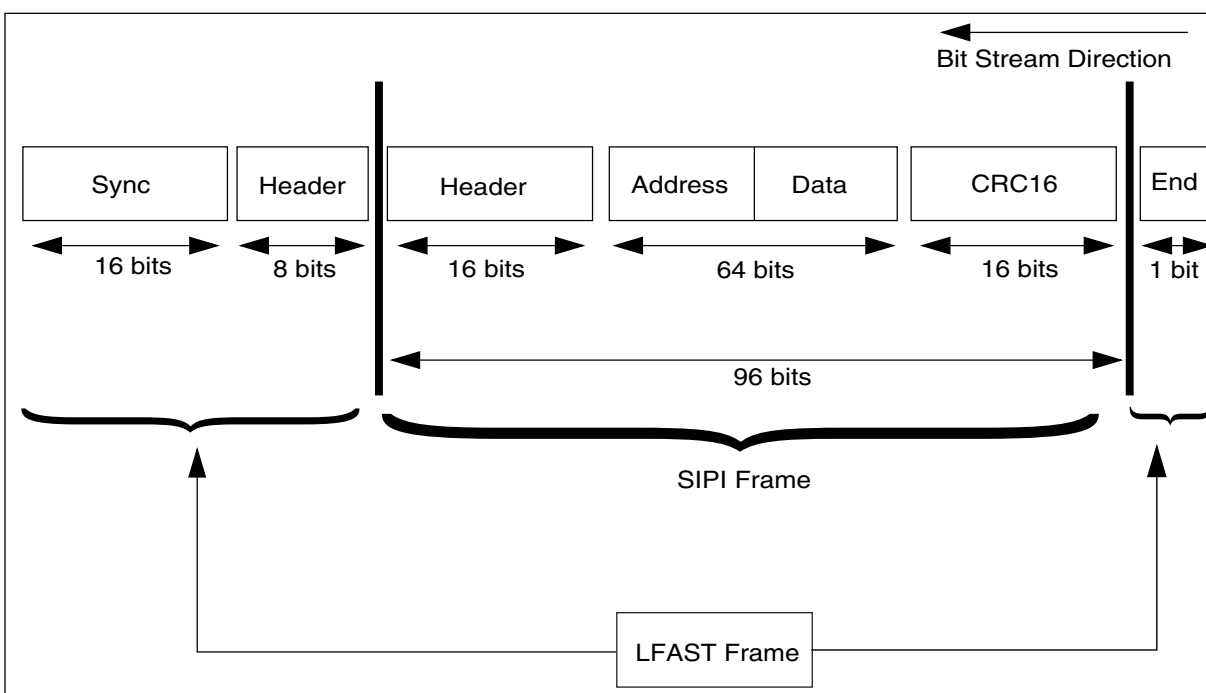


Figure 44-14. Register write transfer (showing LFAST frame encapsulation)

44.8.3.1 Normal write transfer

A normal write transfer contains header, address, data (32-bit) and CRC as shown in [Figure 44-15](#).

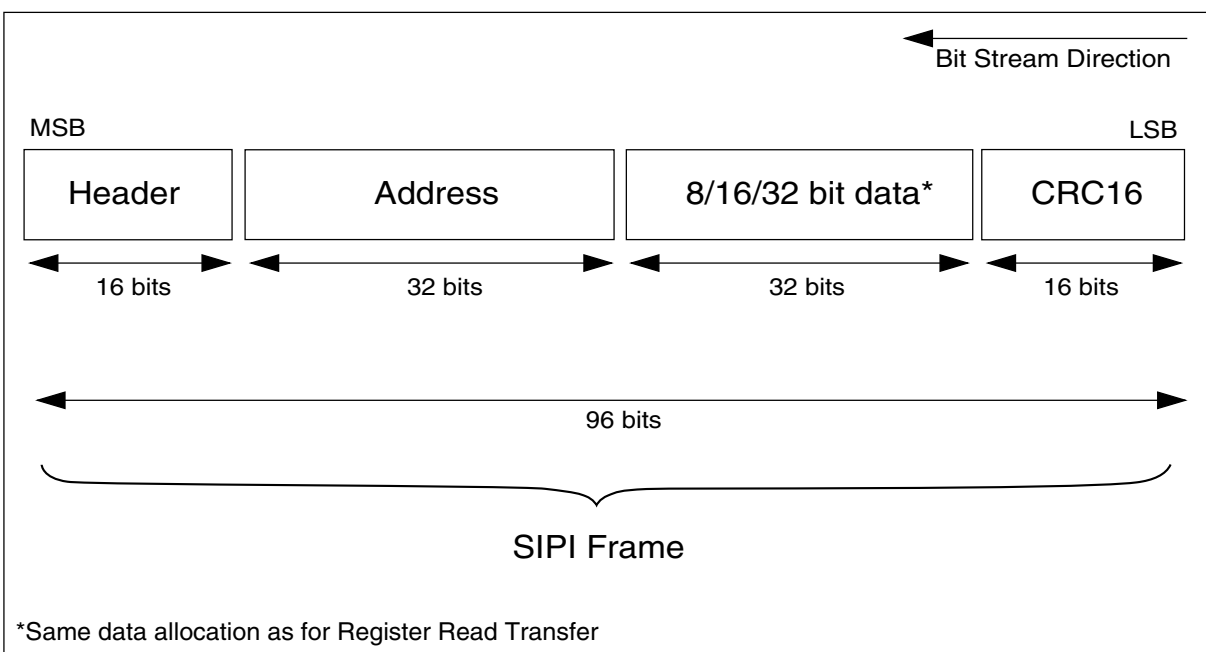


Figure 44-15. Write transfer (LFAST frame encapsulation is not shown)

Note

- For an 8-bit transfer, address bits [31:2] are used as address and bits [1:0] are used as byte enables.
 - Bits[1:0]:
 - 00 - byte 3 enabled (MSB)
 - 01 - byte 2 enabled
 - 10 - byte 1 enabled
 - 11 - byte 0 enabled (LSB)
- For a 16-bit transfer, address bits [31:1] are used as address and bit [0] is used as half word enable.
 - Bit[0]:
 - 0 - half-word 1 enabled (MSB)
 - 1 - half-word 0 enabled (LSB)

44.8.3.2 Streaming write transfer

Streaming write transfer has 256 bits of payload.

44.8.3.2.1 Streaming write transfer with address

Setting the address is performed using a direct write transfer. The SIPI Maximum Count Register (SIPI_MAXCR) and SIPI Address Reload Register (SIPI_ARR) are written before the SIPI Address Count Register (SIPI_ACR). This is to avoid unwanted behavior of the SIPI attempting to access non-shared memory. This is only true for the very first streaming command, subsequent streaming command(s) may not need to write the SIPI_MAXCR and SIPI_ARR (see [SIPI Max Count Register \(SIPI_MAXCR\)](#), [SIPI Address Count Register \(SIPI_ACR\)](#) and [SIPI Address Reload Register \(SIPI_ARR\)](#)).

44.8.3.2.2 Streaming transfer with data

[Figure 44-16](#) shows the packet structure of the streaming transfer containing data.

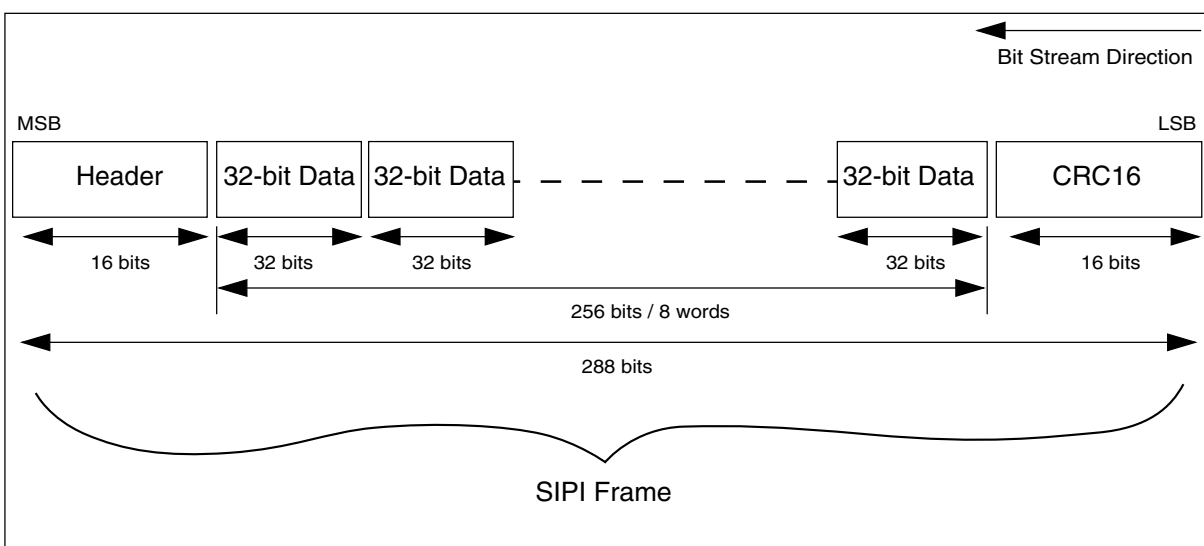


Figure 44-16. Streaming transfer with data

44.8.4 Write Acknowledge transfer

A Write Acknowledge transfer contains only header and CRC bits (see [Write Acknowledge transfer](#)). The CRC bits are calculated on the header field.

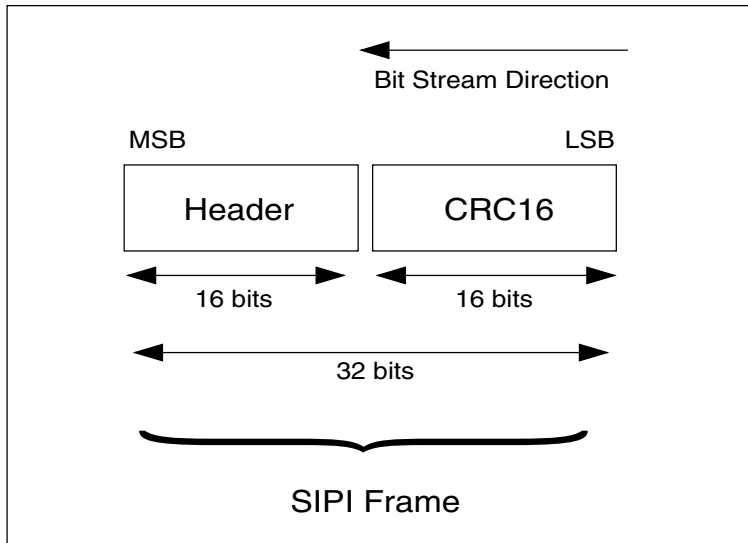


Figure 44-17. Write Acknowledge transfer (LFAST encapsulation is not shown)

44.8.5 ID request response

44.8.5.1 ID request transfer

An ID request is transmitted by the initiator node as shown in [Figure 44-18](#).

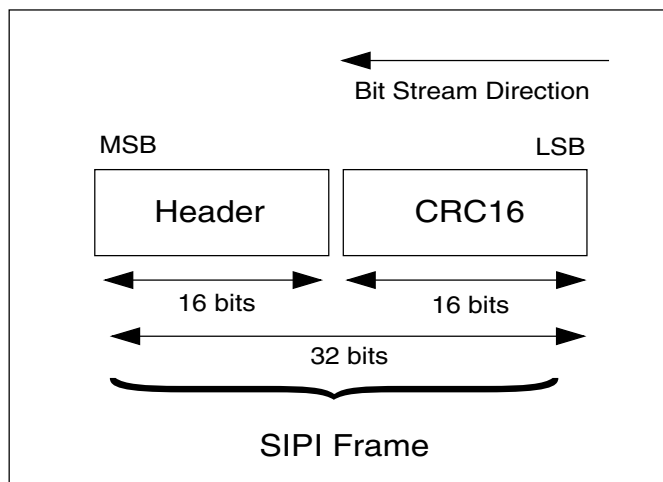


Figure 44-18. ID request transfer (LFAST encapsulation is not shown)

44.8.5.2 ID request response transfer

An ID request response is transmitted by the target node as shown in [Figure 44-19](#).

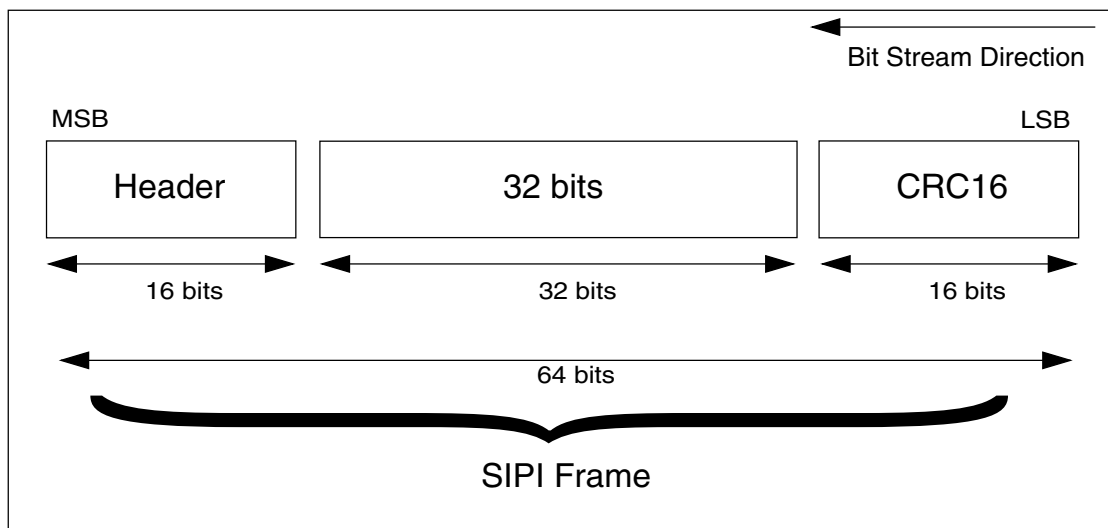
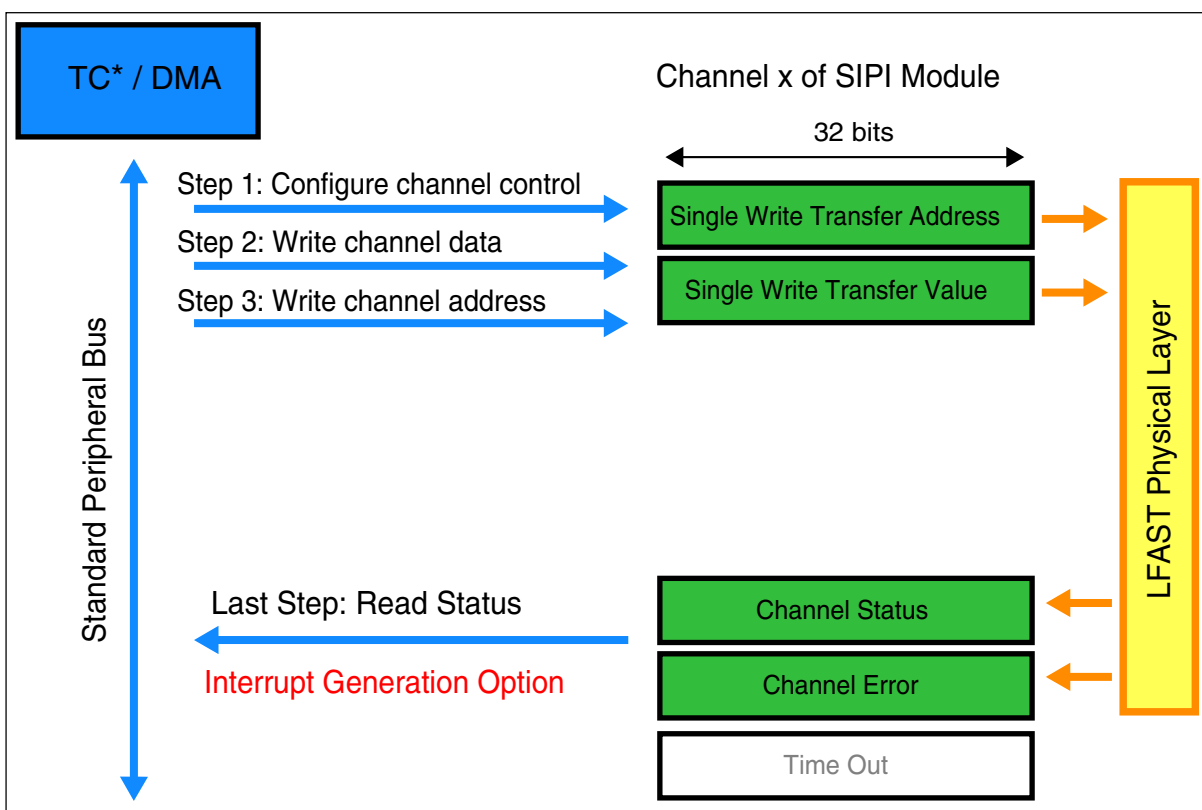


Figure 44-19. ID request response transfer

Note

The ID request response contains the value of the CHIP ID in place of data.

44.9 Transfer API and flow charts



*Note: TC = Transmit Command

Figure 44-20. SIPI single register write API

Implement the following steps to generate a single Write Transfer Request (Figure 44-20):

1. Configure data and `SIPI_CCRn` at the initiator node.
2. Configure `SIPI_CARn` at the initiator node.

As soon as the channel address register (`SIPI_CARn`) is written and if `CCRn[CHEN] = 1`, the initiator SIPI will calculate the CRC on header, address and data field and will start transmitting data to LFAST.

3. Software polls the status register bits (`SIPI_CSRn`) to determine when the request completes, `SIPI_CSRn[ACKR] = 1`. An interrupt will be generated if the corresponding `SIPI_CIRn[ACKIE] = 1`.

On a single Write transfer request reception (Figure 44-21):

1. Target node will place the address, data and control information on its AHB master interface.
2. When the process is completed, the target node will generate an acknowledge frame and send it back to the LFAST.

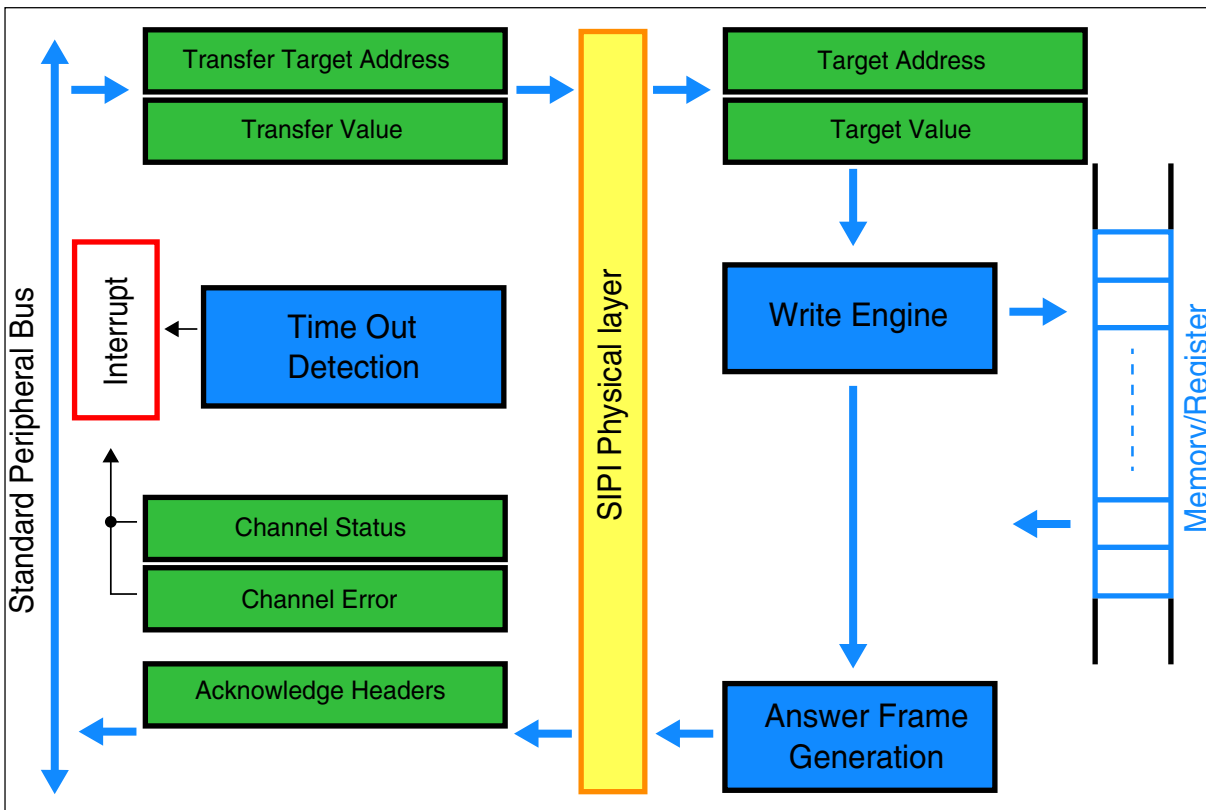


Figure 44-21. SIPI Single Register Write API – Flow Chart

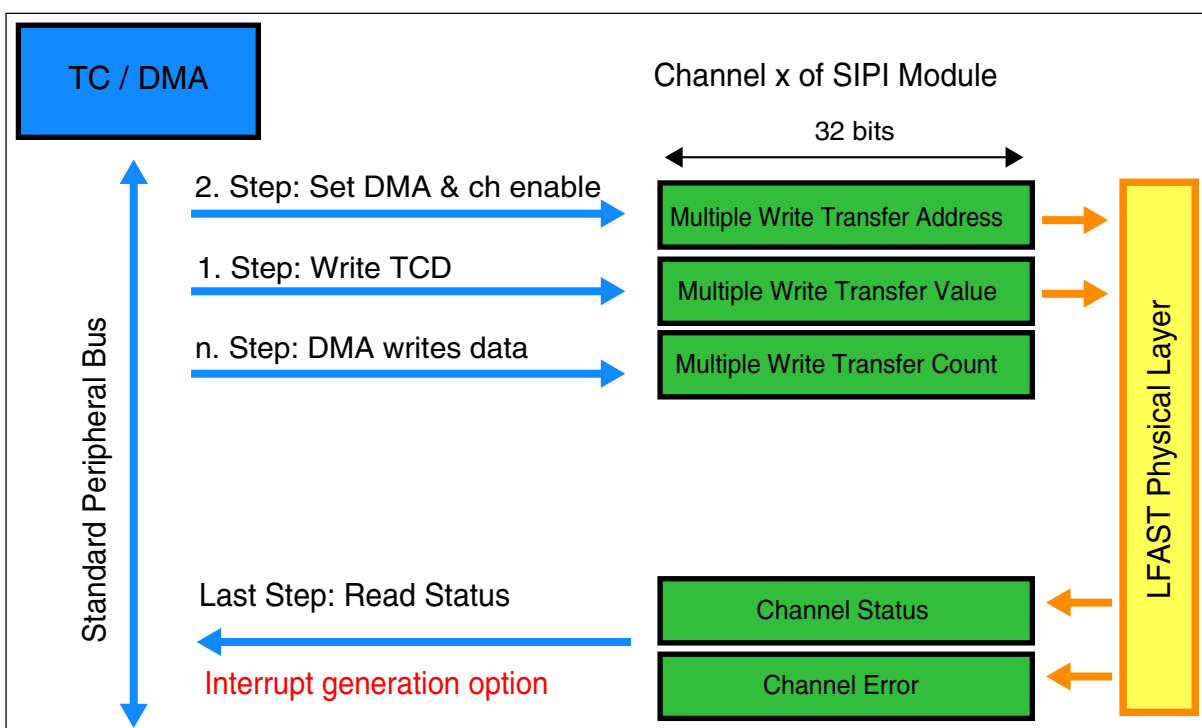


Figure 44-22. SIPI Multiple Register Write API

For multiple write transfer request generation (Figure 44-22):

1. Software will configure the Transfer Control Descriptor (TCD) of the DMA.
2. Software will write `SIPI_CCRn[CHEN] = 1` and `SIPI_CCRn[DAN] = 1`.
3. SIPI will start copying data into the `SIPI_CDRn` through its DMA interface, depending on the transfer count and data registers size. `SIPI_CDR2_0` should be written with the MSB.
4. When the copying process is complete, initiator SIPI will calculate CRC on header, address and data field and start transmitting data to LFAST.
5. Software should poll the `SIPI_CSRn` status register bits to determine if the request has completed. If `SIPI_CSRn[ACKR] = 1`, an interrupt will be generated (if the corresponding `SIPI_CIRn[ACKIE] = 1`).
6. If `SIPI_CCRn[DEN] = 1`, the SIPI request will transfer to the DMA controller. If not, the state machine goes idle.
7. Steps 4–7 will be repeated.

On Multiple Write transfer request reception (Figure 44-22):

1. Target node will place the address, data and control information on its AHB Master Interface.
2. When the process is completed, target node will generate an acknowledge frame and send it back to the LFAST.

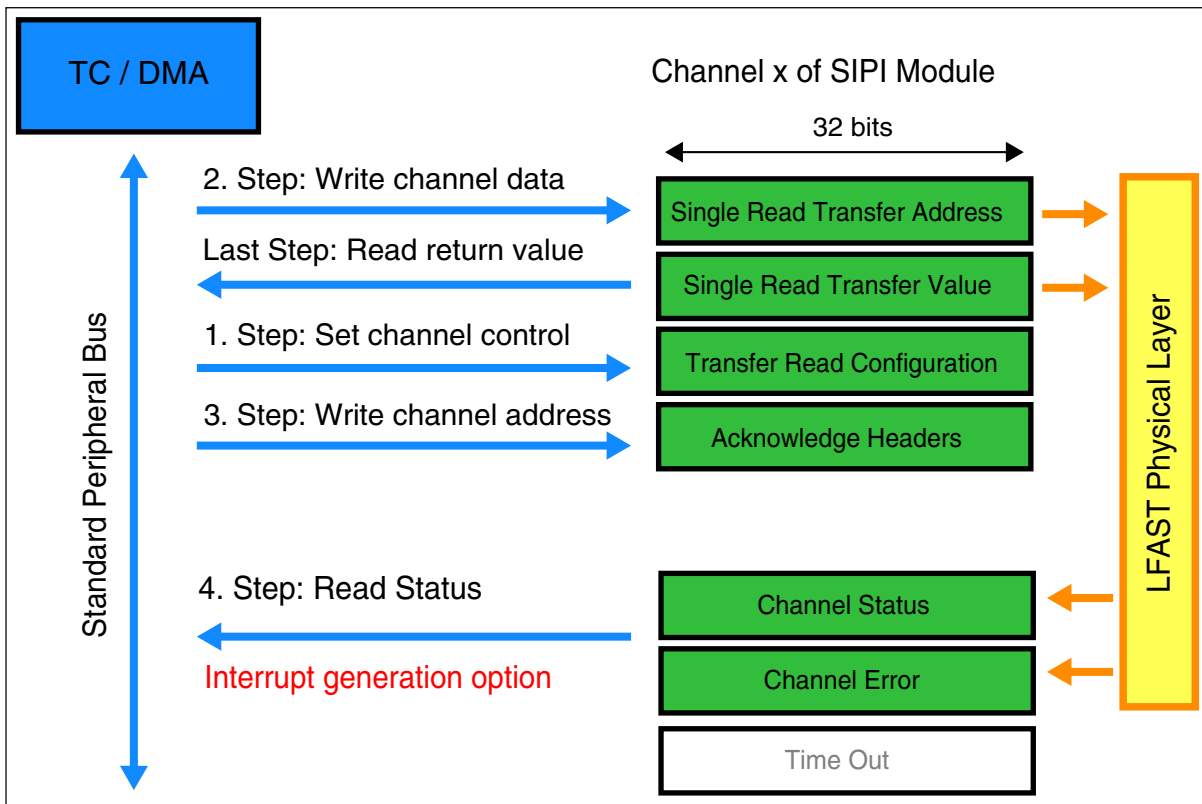


Figure 44-23. SIPI Single Register Read API

For Single Read Transfer Request generation (Figure 44-23):

1. Software/DMA will configure SIPI_CCR_n and SIPI_CDR_n , with the last step by writing CAR_n .
2. As soon as SIPI_CAR is written, the initiator SIPI will calculate CRC of the header and address fields and start transmitting data to the LFAST.
3. Software should poll CSR_n to determine when the request has completed. If $\text{SIPI\_CSR}_n[\text{RAR}] = 1$, an interrupt will be generated (if $\text{SIPI\_CIR}_n[\text{RAIE}] = 1$). If $\text{SIPI\_CSR}_n[\text{RAR}]$ does not set then $\text{SIPI\_ERR}[\text{TOEn}] = 1$ which indicates a timeout has occurred.
4. If $\text{SIPI\_CSR}_n[\text{RAR}] = 1$ then software can read the data register, or can other necessary action if $\text{SIPI\_CSR}_n[\text{RAR}] = 0$.

On Single Read transfer request reception (Figure 44-23):

1. Target node will place the address and control information on its AHB Master Interface.
2. When the process is completed, target node will send read response back to LFAST.

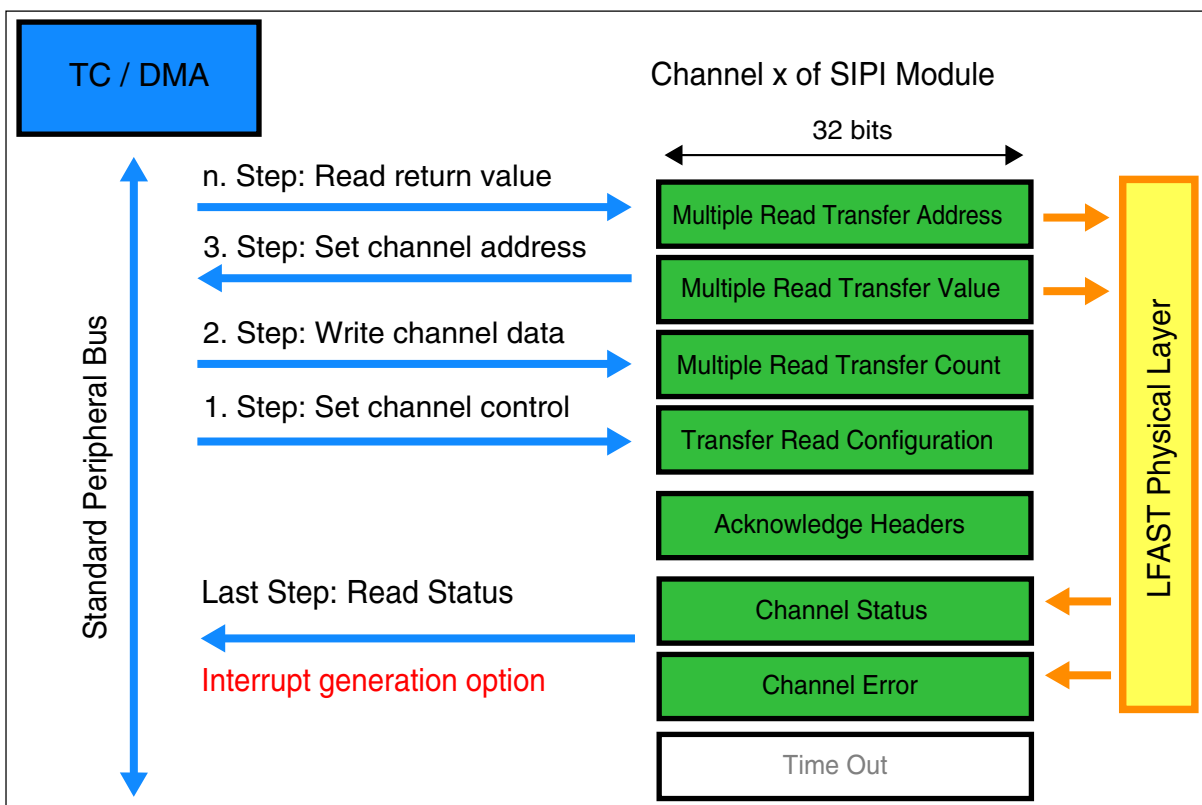


Figure 44-24. SIPI multiple register read API

For multiple read transfer request generation (Figure 44-24):

1. Software/DMA will configure SIPI_CCR_n and SIPI_CDR_n, with the last step by writing SIPI_CAR_n.
2. As soon as channel address register is written, initiator SIPI will calculate CRC on the header and address fields, then starts transmitting data to the LFAST.
3. Software should poll SIPI_CSR_n to determine when the request has completed. If SIPI_CSR_n[RAR] = 1 and SIPI_CIR_n[RAIE] = 1 an interrupt will be generated. If SIPI_CSR_n[RAR] does not set, then SIPI_ERR[TOEn] = 1 which indicates a timeout.
4. Steps 2–3 will be repeated for multiple requests.

On Multiple Read transfer request reception (Figure 44-24):

1. Target node then will start reading data through its AHB interface.
2. When the transfer is completed/all data registers are full, it will calculate CRC and send the response frame back to LFAST.

NOTE

Read answer interrupt pins are also cleared by hardware in DMA transfers. This can lead the software in trouble where it tracks the requests by keep counting the number of times the interrupt was received and cleared.

Figure 44-25, Figure 44-26, Figure 44-27 and Figure 44-28 show the SIPI data streaming of data.

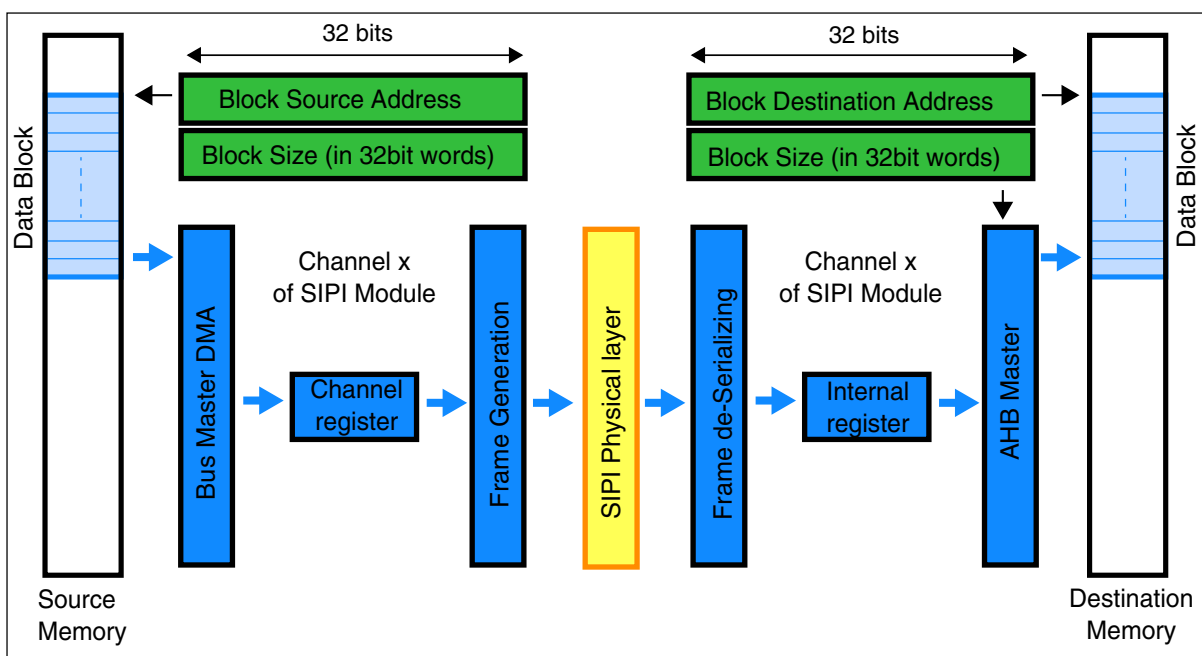


Figure 44-25. SIPI data stream model

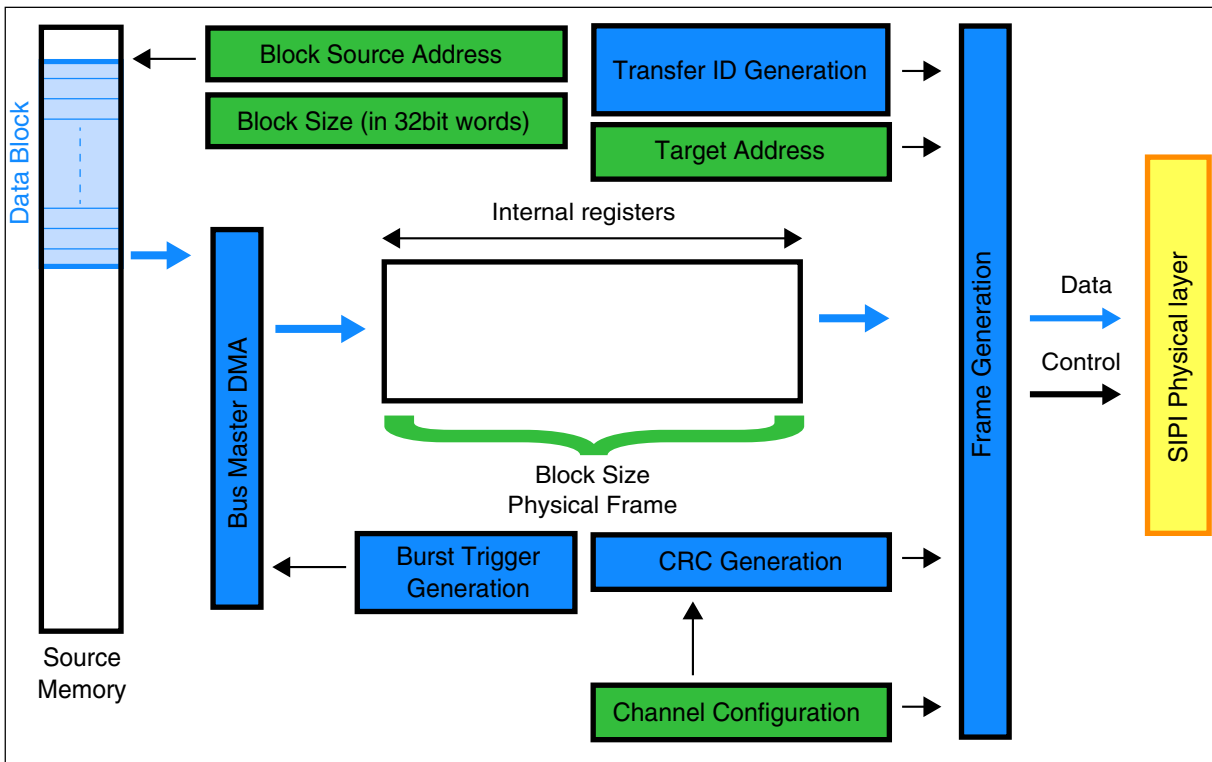


Figure 44-26. SIPI data stream model - Initiator

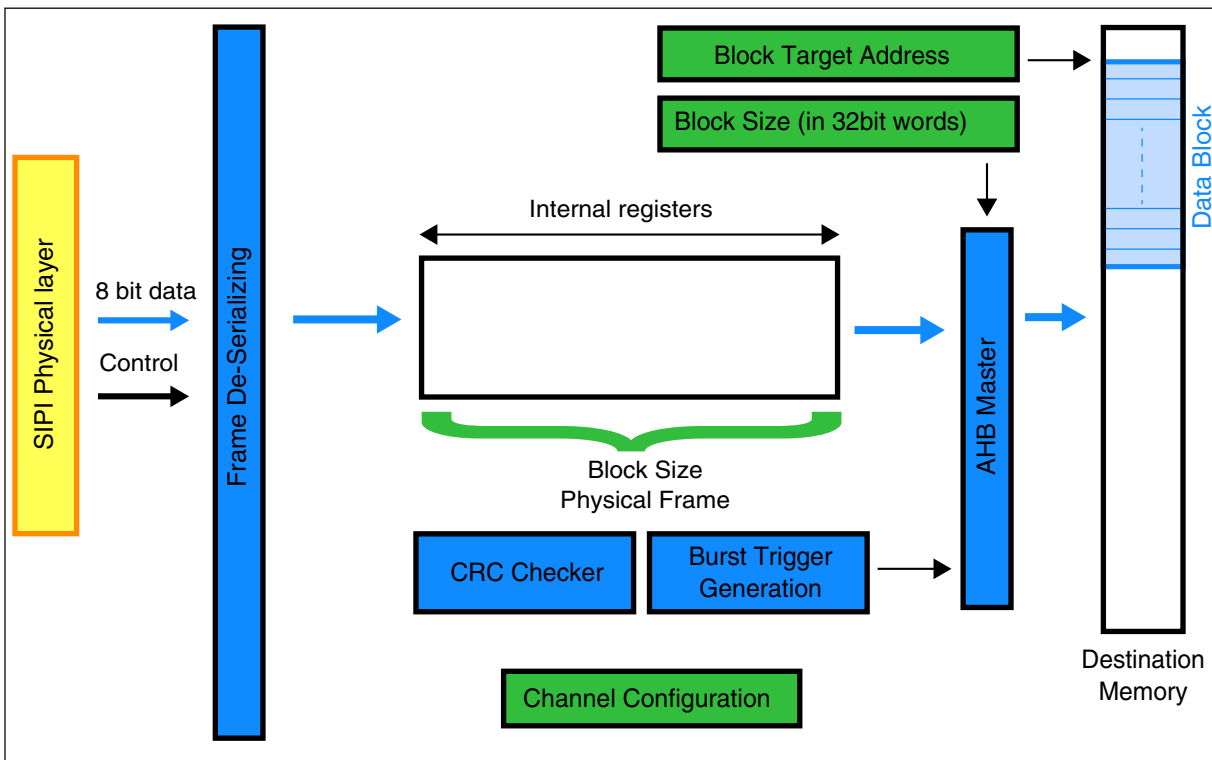


Figure 44-27. SIPI data stream model - Target

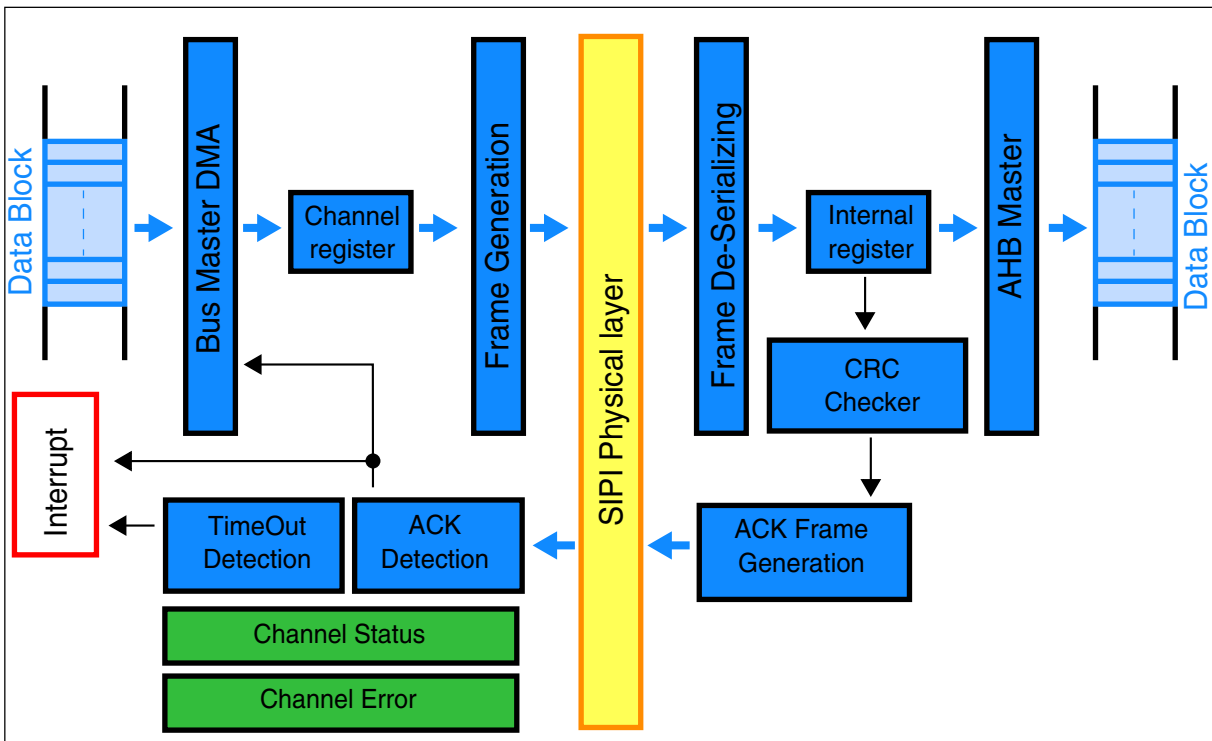


Figure 44-28. SIPI data stream with acknowledge trigger

44.10 DMA programming sequence

The DMA programming sequence is as follows:

1. Software configures TCD (DMA data).
2. DMA transfers from RAM to SIPI registers.
3. SIPI starts transferring the data through LFAST Tx ports.
4. If an acknowledge or fault response is received, SIPI will repeat the process. If any error response is received, SIPI will move to an idle state. An interrupt will be flagged by SIPI, and software will take control of the SIPI.
5. When SIPI receives read response and LFAST's error flags are not asserted, it initiates a DMA request. In case the LFAST asserts error flag, SIPI moves to step 8.
6. When SIPI initiates DMA request, the DMA transfers data between SIPI and RAM.
7. If DMA transfer minor loop/major loop is complete, and the request is negated, ipd_request is negated.
8. If an error occurs:

- The SIPI generates an interrupt
- Software takes control of the SIPI

9. Steps 5 – 8 are repeated until the transfer is finished.

44.11 Modes of operation

SIPI has three operating modes:

- Initialization
- Normal
- Module Disable

After hardware reset the SIPI is in module disable mode, which helps reduce power consumption.

44.11.1 Initialization mode

To enter Initialization mode software writes SIPI_MCR[INIT]=1 (SIPI_MCR[MOEN] must be set before attempting to write SIPI_MCR[INIT]). To exit Initialization mode software writes SIPI_MCR[INIT]=0 (see [SIPI Module Configuration Register \(SIPI_MCR\)](#)).

Note

It is recommended that software checks the state of SIPI (SIPI_MCR[MOEN]) before setting SIPI_MCR[INIT].

44.11.2 Normal mode

Once software has completed initialization of SIPI (SIPI_MCR[INIT]=1), it can enter Normal mode by writing SIPI_MCR[INIT]=0. SIPI needs to be in Normal mode for all data transfers (see [SIPI Module Configuration Register \(SIPI_MCR\)](#)).

Note

SIPI must be enabled before attempting to write SIPI_MCR[INIT] (SIPI_MCR[MOEN]=1).

44.11.3 Module Disable (MD)

MD mode in the SIPI is used to help reduce power consumption. By default, SIPI is in Disable mode, SIPI_MCR[MOEN]=0, and is exited by writing SIPI_MCR[MOEN]=1 (see [SIPI Module Configuration Register \(SIPI_MCR\)](#)). Future SIPI Tx transfers are disabled in Module Disable mode. To disable Rx functionality, the Target Enable (TEN) bit field has to be deasserted, then the internal Rx state machine resets to initial state.

44.12 Errors

This section describes the potential errors that can occur during SIPI operation.

44.12.1 Timeout error

A timeout error is generated at the initiator node when the acknowledge/response is not received within the time configured in the corresponding CTOR_n[TOR] field setting (see [SIPI Channel Timeout Register 0 \(SIPI_CTOR0\)](#), [SIPI Channel Timeout Register 1 \(SIPI_CTOR1\)](#), [SIPI Channel Timeout Register 2 \(SIPI_CTOR2\)](#) and [SIPI Channel Timeout Register 3 \(SIPI_CTOR3\)](#)). A timeout error is indicated when ERR[TOEn]=1 (see [SIPI Error Register \(SIPI_ERR\)](#)).

Note

SIPI should not drop the response even after a timeout occurs. Software will poll both error and status flags after the transfer to see if there was a timeout error. If there was a timeout error the response received may then be discarded.

44.12.2 CRC error

A CRC error is generated at the target nodes when the CRC received with the frame does not match the calculated CRC, and the SR[GCRCE] is set (see [SIPI Status Register \(SIPI_SR\)](#)). An interrupt will be asserted if MCR[CRCIE]=1 (see [SIPI Module Configuration Register \(SIPI_MCR\)](#)).

Note

The target node will not send an acknowledge to the initiator node when a CRC error is generated. An interrupt will be generated on the target side if the corresponding interrupt

enable bit is set. The initiator node will detect a timeout and take necessary action.

44.12.3 Maximum count reached error

The maximum count reached error is only generated at the target node. It is generated when the value of the SIPI_ACR is equal to the SIPI_MAXCR (see [SIPI Max Count Register \(SIPI_MAXCR\)](#) and [SIPI Address Count Register \(SIPI_ACR\)](#)). When the maximum count is reached, SIPI_SR[MCR] = 1 (see [SIPI Status Register \(SIPI_SR\)](#)). An interrupt will be generated if SIPI_MCR[MCRIE] = 1 (see [SIPI Module Configuration Register \(SIPI_MCR\)](#)).

44.12.4 Transaction ID error

The Transaction ID (TID) error is always generated at the initiator node only. It is generated when header bits 15–13 do not match SIPI_CSR n [TID] (transaction ID bits, see [SIPI Channel Status Register 0 \(SIPI_CSR0\)](#), [SIPI Channel Status Register 1 \(SIPI_CSR1\)](#), [SIPI Channel Status Register 2 \(SIPI_CSR2\)](#), and [SIPI Channel Status Register 3 \(SIPI_CSR3\)](#)). SIPI_CSR n [TIDE] = 1 when a TID error is detected, and an interrupt will be generated if SIPI_CIR n [TIDIE] = 1 (see [SIPI Channel Interrupt Register 0 \(SIPI_CIR0\)](#), [SIPI Channel Interrupt Register 1 \(SIPI_CIR1\)](#), [SIPI Channel Interrupt Register 2 \(SIPI_CIR2\)](#), and [SIPI Channel Interrupt Register 3 \(SIPI_CIR3\)](#)).

44.12.5 Acknowledge error

An incorrect acknowledge is received only from the initiator. When the acknowledge received is incorrect, SIPI_ERR[ACKR n] will be set (see [SIPI Error Register \(SIPI_ERR\)](#)). An interrupt will be generated if SIPI_CIR n [WAIE]=1 (see [SIPI Channel Interrupt Register 0 \(SIPI_CIR0\)](#), [SIPI Channel Interrupt Register 1 \(SIPI_CIR1\)](#), [SIPI Channel Interrupt Register 2 \(SIPI_CIR2\)](#), and [SIPI Channel Interrupt Register 3 \(SIPI_CIR3\)](#)).

44.13 CRC calculation

Example: If header is AABBh, address is 1122_3344h and data is CCDD_EEFFh. Then CRC calculation will take place as follows:

- 1) The CRC seed is initialized by FFFF_FFFFh.

- 2) All the data will be mirrored before sending to CRC engine (for example, MSB will be sent as LSB).
- 3) So the header will be sent as DD55_0000h.
- 4) Address will be sent as 22CC_4488h.
- 5) Data will be sent as FF77_BB33h.

44.14 Interrupt logic

A description of the interrupt logic can be found in the following figure.

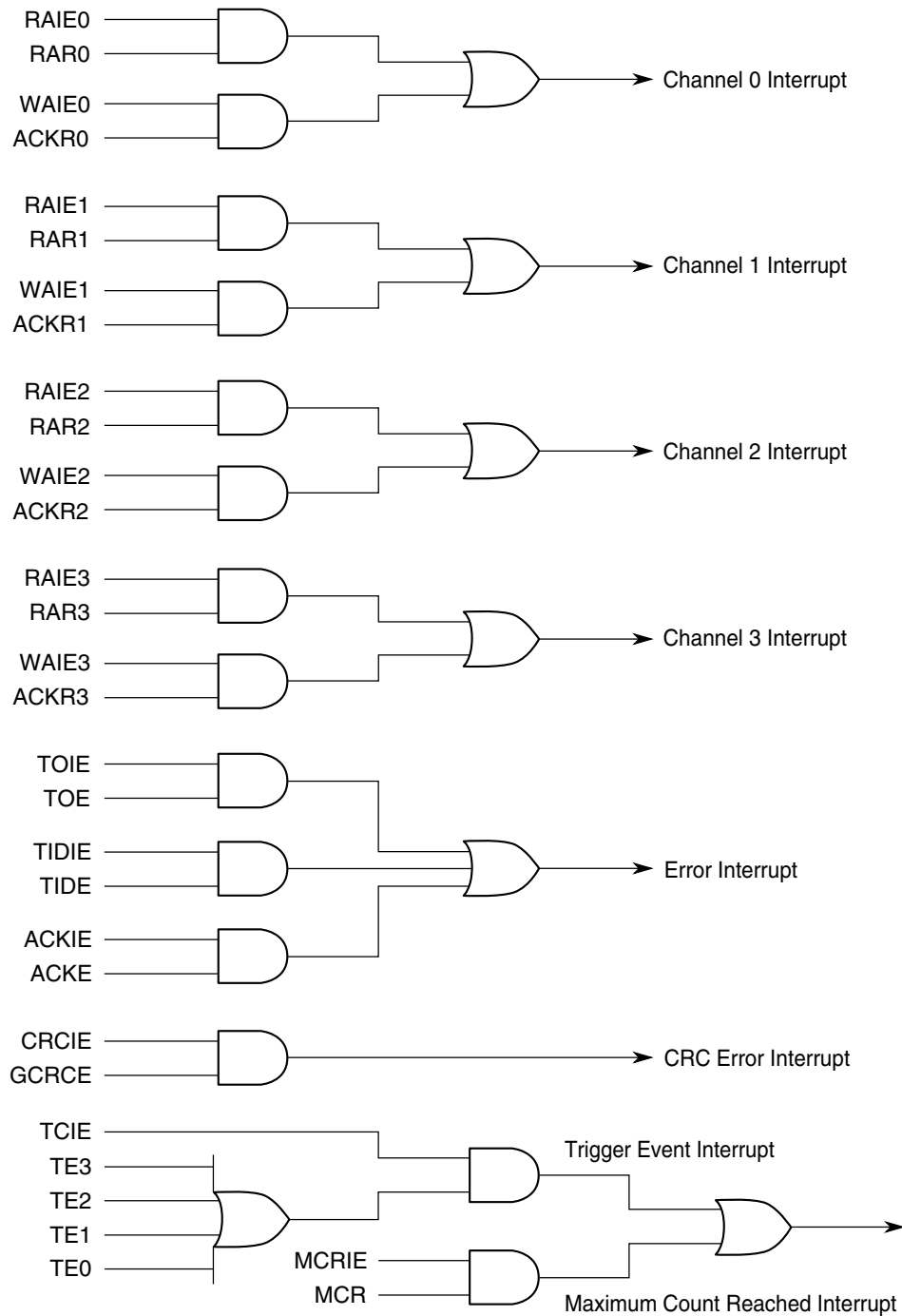


Figure 44-29. Interrupt description

44.15 SIPI control and status overview

The diagram below shows the relationship between transfers and the SIPI control and status registers.

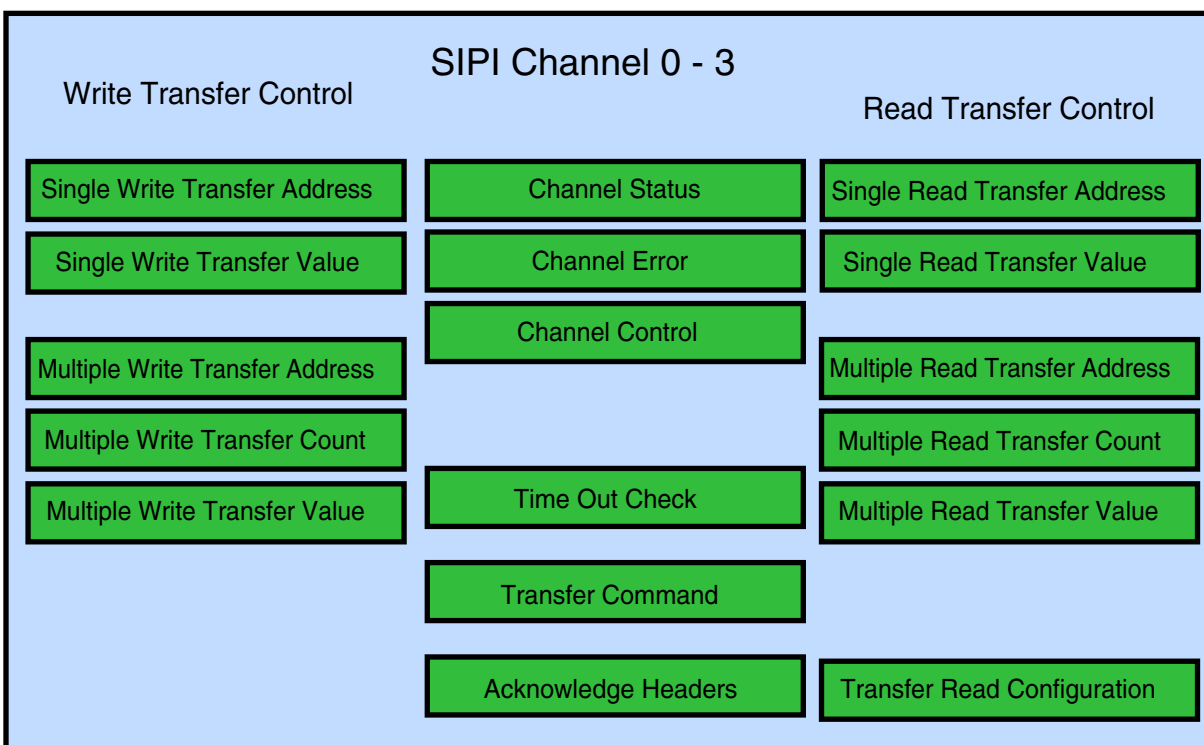


Figure 44-30. SIPI control and status overview – Register transfer

44.16 Memory map and register definition

SIPI memory map

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
0	SIPI Channel Control Register 0 (SIPI_CCR0)	32	R/W	0000_0000h	44.16.1/ 2271
4	SIPI Channel Status Register 0 (SIPI_CSR0)	32	R	0000_0000h	44.16.2/ 2274
C	SIPI Channel Interrupt Register 0 (SIPI_CIR0)	32	R/W	0000_0000h	44.16.3/ 2275
10	SIPI Channel Timeout Register 0 (SIPI_CTOR0)	32	R/W	0000_00FFh	44.16.4/ 2276
14	SIPI Channel CRC Register 0 (SIPI_CCRC0)	32	R	0000_0000h	44.16.5/ 2277
18	SIPI Channel Address Register 0 (SIPI_CAR0)	32	R/W	0000_0000h	44.16.6/ 2277
1C	SIPI Channel Data Register 0 (SIPI_CDR0)	32	R/W	0000_0000h	44.16.7/ 2278
20	SIPI Channel Control Register 1 (SIPI_CCR1)	32	R/W	0000_0000h	44.16.8/ 2278

Table continues on the next page...

SIPI memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
24	SIPI Channel Status Register 1 (SIPI_CSR1)	32	R	0000_0000h	44.16.9/2281
2C	SIPI Channel Interrupt Register 1 (SIPI_CIR1)	32	R/W	0000_0000h	44.16.10/2283
30	SIPI Channel Timeout Register 1 (SIPI_CTOR1)	32	R/W	0000_00FFh	44.16.11/2284
34	SIPI Channel CRC Register 1 (SIPI_CCRC1)	32	R	0000_0000h	44.16.12/2285
38	SIPI Channel Address Register 1 (SIPI_CAR1)	32	R/W	0000_0000h	44.16.13/2285
3C	SIPI Channel Data Register 1 (SIPI_CDR1)	32	R/W	0000_0000h	44.16.14/2286
40	SIPI Channel Control Register 2 (SIPI_CCR2)	32	R/W	0000_0000h	44.16.15/2286
44	SIPI Channel Status Register 2 (SIPI_CSR2)	32	R	0000_0000h	44.16.16/2289
4C	SIPI Channel Interrupt Register 2 (SIPI_CIR2)	32	R/W	0000_0000h	44.16.17/2291
50	SIPI Channel Timeout Register 2 (SIPI_CTOR2)	32	R/W	0000_00FFh	44.16.18/2292
54	SIPI Channel CRC Register 2 (SIPI_CCRC2)	32	R	0000_0000h	44.16.19/2293
58	SIPI Channel Address Register 2 (SIPI_CAR2)	32	R/W	0000_0000h	44.16.20/2293
5C	SIPI Channel Data Register 2 (SIPI_CDR2_0)	32	R/W	0000_0000h	44.16.21/2294
60	SIPI Channel Data Register 2 (SIPI_CDR2_1)	32	R/W	0000_0000h	44.16.21/2294
64	SIPI Channel Data Register 2 (SIPI_CDR2_2)	32	R/W	0000_0000h	44.16.21/2294
68	SIPI Channel Data Register 2 (SIPI_CDR2_3)	32	R/W	0000_0000h	44.16.21/2294
6C	SIPI Channel Data Register 2 (SIPI_CDR2_4)	32	R/W	0000_0000h	44.16.21/2294
70	SIPI Channel Data Register 2 (SIPI_CDR2_5)	32	R/W	0000_0000h	44.16.21/2294
74	SIPI Channel Data Register 2 (SIPI_CDR2_6)	32	R/W	0000_0000h	44.16.21/2294
78	SIPI Channel Data Register 2 (SIPI_CDR2_7)	32	R/W	0000_0000h	44.16.21/2294
7C	SIPI Channel Control Register 3 (SIPI_CCR3)	32	R/W	0000_0000h	44.16.22/2294
80	SIPI Channel Status Register 3 (SIPI_CSR3)	32	R	0000_0000h	44.16.23/2297

Table continues on the next page...

SIPI memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
88	SIPI Channel Interrupt Register 3 (SIPI_CIR3)	32	R/W	0000_0000h	44.16.24/2299
8C	SIPI Channel Timeout Register 3 (SIPI_CTOR3)	32	R/W	0000_00FFh	44.16.25/2300
90	SIPI Channel CRC Register 3 (SIPI_CCRC3)	32	R	0000_0000h	44.16.26/2301
94	SIPI Channel Address Register 3 (SIPI_CAR3)	32	R/W	0000_0000h	44.16.27/2301
98	SIPI Channel Data Register 3 (SIPI_CDR3)	32	R/W	0000_0000h	44.16.28/2302
9C	SIPI Module Configuration Register (SIPI_MCR)	32	R/W	See section	44.16.29/2302
A0	SIPI Status Register (SIPI_SR)	32	R	0000_0000h	44.16.30/2305
A4	SIPI Max Count Register (SIPI_MAXCR)	32	R/W	FFFF_FFFCh	44.16.31/2307
A8	SIPI Address Reload Register (SIPI_ARR)	32	R/W	0000_0000h	44.16.32/2307
AC	SIPI Address Count Register (SIPI_ACR)	32	R/W	0000_0000h	44.16.33/2308
B0	SIPI Error Register (SIPI_ERR)	32	R	0000_0000h	44.16.34/2309

44.16.1 SIPI Channel Control Register 0 (SIPI_CCR0)

NOTE

Back to back transactions are not supported on a single channel (for example, if the channel busy bit, SIPI_CSR0[CB] = 1, is configured for a channel, it cannot be triggered for another transfer until the data buffers are empty and the channel is free, SIPI_CSR0[CB] = 0).

PRIORITY SCHEDULING: The channel whose SIPI_CAR_n is written first will gain access first irrespective of its priority at the start of the transfer. As more transfers are requested, priority will be resolved as per the priority scheme (for example, channel 0 – channel 1 – channel 2 – channel 3). Scheduling will occur each time the LFAST is not ready to receive data and no channel is pending for data transmission. The reason for the scheduling is that a common CRC module is shared by all channels. If the LFAST is ready to receive data during the entire process, it will happen only in the beginning and later on priority will be resolved as per the priority scheme.

NOTE

This register is only writable in Initialization mode (SIPI_MCR[INIT] = 1), except for command bits (for example, TC, WL, ST, IDT, RRT and WRT). Also, the command bits can be written only when the corresponding Channel busy bit is not asserted (SIPI_CSR_n[CB] = 1 (see [SIPI Channel Status Register 0 \(SIPI_CSR0\)](#)). Software will need to poll the respective channel busy bits before attempting to change the command type.

Only one channel at a time can transmit. Therefore, only one channel can remain busy. Maximum time allowed to transmit a full command is nine clock cycles. The Command type of a channel can only be changed when it is no longer busy. Also, to send a command, we need to write to the corresponding SIPI_CAR_n (see [SIPI Channel Address Register 0 \(SIPI_CAR0\)](#)).

Memory map and register definition

Address: 0h base + 0h offset = 0h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0														TC	
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0								WL		CHEN	ST	IDT	RRT	WRT	DEN
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

SIPI_CCR0 field descriptions

Field	Description
0–14 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
15 TC	Send Trigger Command. A Trigger Command requires that only this bit to be set by software, the trigger does not depend on settings in SIPI_CAR0. 0 Trigger command not sent 1 Trigger command sent
16–23 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
24–25 WL	Word Length Transfer. For Streaming write, WL bits should be written 10. 00 8-bit 01 16-bit 10 32-bit 11 not used
26 CHEN	Channel Enable. If all channel enables are simultaneously written to enable the channels, channel 0 will be given the highest priority for transmission. For channels, the lowest channel number is given highest priority. 0 Channel is disabled 1 Channel is enabled
27 ST	Streaming Transfer. This bit is hard-coded to 0. It can only be written for channel 2 (SIPI_CCR2). This bit can only be written when the corresponding SIPI_CCRn[WRT] = 1. NOTE: Only transfers with channel 2 can write 1 to this bit. All other channels force SIPI_CCRn[ST] = 0. The SIPI_CCR2[WRT] and SIPI_CCR2[ST] bits must be set for streaming.

Table continues on the next page...

SIPI_CCR0 field descriptions (continued)

Field	Description
	0 Streaming transfer is disabled 1 Streaming transfer is enabled
28 IDT	ID Read Request Transfer. This request returns the value of the CHIP ID. NOTE: The order of priority for writing to IDT, WRT and RRT must be in this order: 1. IDT 2. RRT 3. WRT 0 ID read request not sent 1 ID read request sent
29 RRT	Read Request Transfer. NOTE: The order of priority for writing to IDT, WRT and RRT must be in this order: 1. IDT 2. RRT 3. WRT 0 Read request will not be sent 1 Read request transfer by the initiator. This bit can not be written if SIPI_CCR0[IDT] = 1.
30 WRT	Write Request Transfer. NOTE: The order of priority for writing to IDT, WRT and RRT must be in this order: 1. IDT 2. RRT 3. WRT 0 No write request will be sent 1 Write request transfer by the initiator. This bit can not be written if SIPI_CCR0[IDT] = 1 or SIPI_CCR0[RRT] = 1.
31 DEN	DMA Enable. When enabling DMA mode, this bit should be set by software. It will then be cleared by hardware after one major loop has completed. 0 Channel will be used for bus interface access. 1 Channel will be used for DMA access

44.16.2 SIPI Channel Status Register 0 (SIPI_CSR0)

CSR0 contains the status bits for the current transfer.

Address: 0h base + 4h offset = 4h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0								RAR	TID			ACKR	CB	0	
W									w1c				w1c			
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

SIPI_CSR0 field descriptions

Field	Description
0–23 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
24 RAR	Read Answer Reception. 0 Read answer not received 1 Read answer received
25–27 TID	Transaction ID of transmitted frame. Transaction ID is a random number associated with every Tx frame to match the received response/ack with the command.
28 ACKR	Acknowledge Received. 0 Acknowledge not received 1 Acknowledge received
29 CB	Channel Busy. Indicates channel 0 status. 0 Channel 0 free 1 Channel 0 busy

Table continues on the next page...

SIPI_CSR0 field descriptions (continued)

Field	Description
30–31 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

44.16.3 SIPI Channel Interrupt Register 0 (SIPI_CIR0)

SIPI_CIR0 contains the interrupt enable bits for channel 0.

Address: 0h base + Ch offset = Ch

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	Reserved															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	Reserved															
W											WAIE	RAIE	TCIE	TOIE	TIDIE	ACKIE
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

SIPI_CIR0 field descriptions

Field	Description
0–25 Reserved	This field is reserved.
26 WAIE	Write Acknowledge Interrupt Enable. 0 Interrupt is disabled 1 Interrupt is enabled
27 RAIE	Read Answer Interrupt Enable. 0 Interrupt is disabled 1 Interrupt is enabled

Table continues on the next page...

SIPI_CIR0 field descriptions (continued)

Field	Description
28 TCIE	Trigger Command Interrupt Enable. 0 Interrupt is disabled 1 Interrupt is enabled
29 TOIE	Timeout Error Interrupt Enabled. 0 Interrupt is disabled 1 Interrupt is enabled
30 TIDIE	Transaction ID Error Interrupt Enable. 0 Interrupt is disabled 1 Interrupt is enabled
31 ACKIE	Acknowledge Error Interrupt Enable. 0 Interrupt is disabled 1 Interrupt is enabled

44.16.4 SIPI Channel Timeout Register 0 (SIPI_CTOR0)

SIPI_CTOR0 contains the timeout value for Tx requests.

Address: 0h base + 10h offset = 10h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0																							TOR								
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1	

SIPI_CTOR0 field descriptions

Field	Description
0–23 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
24–31 TOR	Timeout value for transmitted requests. Timeout counter runs on the prescaled peripheral clock. Prescaler is defined by SIPI_MCR[PRSCLR]. NOTE: Field can only be written during Initialization mode (SIPI_MCR[INIT] = 1).

44.16.5 SIPI Channel CRC Register 0 (SIPI_CCRC0)

SIPI_CCRC n is a read only which returns the CRC calculated value on the header, address and data bits.

Address: 0h base + 14h offset = 14h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	CRCI																CRCT															
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	

SIPI_CCRC0 field descriptions

Field	Description
0–15 CRCI	Reflects received CRC value at initiator
16–31 CRCT	Reflects received CRC value at target

44.16.6 SIPI Channel Address Register 0 (SIPI_CAR0)

SIPI_CAR0 is the address target for data transmission.

Address: 0h base + 18h offset = 18h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	CAR																															
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	

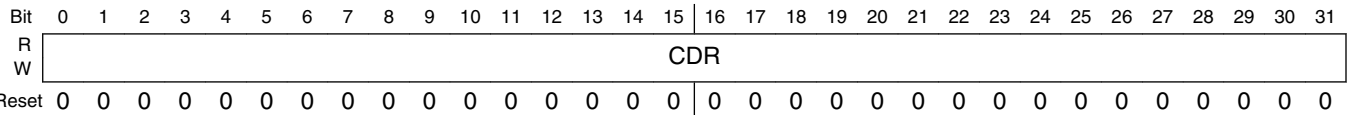
SIPI_CAR0 field descriptions

Field	Description
0–31 CAR	These bits contain the address of the target node.

44.16.7 SIPI Channel Data Register 0 (SIPI_CDR0)

SIPI_CDR0 contains the data to be transmitted, and can be written anytime by software or the DMA. The data can be written in 8, 16 or 32 bit formats.

Address: 0h base + 1Ch offset = 1Ch



SIPI_CDR0 field descriptions

Field	Description
0–31 CDR	Data register bits. Contains the data that will be transmitted, or received.

44.16.8 SIPI Channel Control Register 1 (SIPI_CCR1)

NOTE

Back to back transactions are not supported on a single channel (for example, if the channel busy bit, SIPI_CSR1[CB] = 1, is configured for a channel, it cannot be triggered for another transfer until the data buffers are empty and the channel is free, SIPI_CSR1[CB] = 0).

PRIORITY SCHEDULING: The channel whose SIPI_CAR_n is written first will gain access first irrespective of its priority at the start of the transfer. As more transfers are requested, priority will be resolved as per the priority scheme (for example, channel 0 – channel 1 – channel 2 – channel 3). Scheduling will occur each time the LFAST is not ready to receive data and no channel is pending for data transmission. The reason for the scheduling is that a common CRC module is shared by all channels. If the LFAST is ready to receive data during the entire process, it will happen only in the beginning and later on priority will be resolved as per the priority scheme.

NOTE

This register is only writable in Initialization mode (SIPI_MCR[INIT] = 1), except for command bits (for example, TC, WL, ST, IDT, RRT and WRT). Also, the command bits can be written only when the corresponding Channel busy bit is not asserted (SIPI_CSR n [CB] = 1 (see [SIPI Channel Status Register 1 \(SIPI_CSR1\)](#)). Software will need to poll the respective channel busy bits before attempting to change the command type.

Only one channel at a time can transmit. Therefore, only one channel can remain busy. Maximum time allowed to transmit a full command is nine clock cycles. The Command type of a channel can only be changed when it is no longer busy. Also, to send a command, we need to write to the corresponding SIPI_CAR n (see [SIPI Channel Address Register 1 \(SIPI_CAR1\)](#)).

Address: 0h base + 20h offset = 20h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0								WL		CHEN	ST	IDT	RRT	WRT	DEN
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

SIPI_CCR1 field descriptions

Field	Description
0–14 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
15 TC	Send Trigger Command. A Trigger Command requires that only this bit to be set by software, the trigger does not depend on settings in SIPI_CAR1. 0 Trigger command not sent 1 Trigger command sent

Table continues on the next page...

SIPI_CCR1 field descriptions (continued)

Field	Description
16–23 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
24–25 WL	Word Length Transfer. For Streaming write WL bits should be written 10. 00 8-bit 01 16-bit 10 32-bit 11 not used
26 CHEN	Channel Enable. If all channel enables are simultaneously written to enable the channels, channel 0 will be given the highest priority for transmission. For channels, the lowest channel number is given highest priority. 0 Channel is disabled 1 Channel is enabled
27 ST	This bit is hard-coded to 0. It can only be written for channel 2 (SIPI_CCR2). This bit can only be written when the corresponding SIPI_CCR _n [WRT] = 1. NOTE: Only transfers with channel 2 can write 1 to this bit. All other channels force SIPI_CCR _n [ST] = 0. The SIPI_CCR2[WRT] and SIPI_CCR2[ST] bits must be set for streaming. 0 Streaming transfer is disabled 1 Streaming transfer is enabled
28 IDT	ID Read Request Transfer. This request returns the value of the CHIP ID. NOTE: The order of priority for writing to IDT, WRT and RRT must be in this order: 1. IDT 2. RRT 3. WRT 0 ID read request not sent 1 ID read request sent
29 RRT	Read Request Transfer. NOTE: The order of priority for writing to IDT, WRT and RRT must be in this order: 1. IDT 2. RRT 3. WRT 0 Read request will not be sent 1 Read request transfer by the initiator. This bit can not be written if SIPI_CCR1[IDT] = 1.
30 WRT	Write Request Transfer. NOTE: The order of priority for writing to IDT, WRT and RRT must be in this order: 1. IDT 2. RRT 3. WRT

Table continues on the next page...

SIPI_CCR1 field descriptions (continued)

Field	Description
	0 No write request will be sent 1 Write request transfer by the initiator. This bit can not be written if SIPI_CCR1[IDT] = 1 or SIPI_CCR1[RRT] = 1.
31 DEN	DMA Enable. When enabling DMA mode, this bit should be set by software. It will then be cleared by hardware after one major loop has completed. 0 Channel will be used for bus interface access. 1 Channel will be used for DMA access

44.16.9 SIPI Channel Status Register 1 (SIPI_CSR1)

CSR1 contains the status bits for the current transfer.

Address: 0h base + 24h offset = 24h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0								RAR	TID			ACKR	CB	0	
W									w1c				w1c			
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

SIPI_CSR1 field descriptions

Field	Description
0–23 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
24 RAR	Read Answer Reception.

Table continues on the next page...

SIPI_CSR1 field descriptions (continued)

Field	Description
	0 Read answer not received 1 Read answer received
25–27 TID	Transaction ID of transmitted frame. Transaction ID is a random number associated with every Tx frame to match the received response/ack with the command.
28 ACKR	Acknowledge Received. 0 Acknowledge not received 1 Acknowledge received
29 CB	Channel Busy. Indicates channel 1 status. 0 Channel 1 free 1 Channel 1 busy
30–31 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

44.16.10 SIPI Channel Interrupt Register 1 (SIPI_CIR1)

The CIR1 contains the interrupt enable bits for channel 1.

Address: 0h base + 2Ch offset = 2Ch

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	Reserved															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	Reserved															
W											WAIE	RAIE	TCIE	TOIE	TIDIE	ACKIE
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

SIPI_CIR1 field descriptions

Field	Description
0–25 Reserved	This field is reserved.
26 WAIE	Write Acknowledge Interrupt Enable. 0 Interrupt is disabled 1 Interrupt is enabled
27 RAIE	Read Answer Interrupt Enable. 0 Interrupt is disabled 1 Interrupt is enabled
28 TCIE	Trigger Command Interrupt Enable. 0 Interrupt is disabled 1 Interrupt is enabled
29 TOIE	Timeout Error Interrupt Enabled.

Table continues on the next page...

SIPI_CIR1 field descriptions (continued)

Field	Description
	0 Interrupt is disabled 1 Interrupt is enabled
30 TIDIE	Transaction ID Error Interrupt Enable. 0 Interrupt is disabled 1 Interrupt is enabled
31 ACKIE	Acknowledge Error Interrupt Enable. 0 Interrupt is disabled 1 Interrupt is enabled

44.16.11 SIPI Channel Timeout Register 1 (SIPI_CTOR1)

SIPI_CTOR1 contains the timeout value for Tx requests.

Address: 0h base + 30h offset = 30h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0																							TOR								
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1	

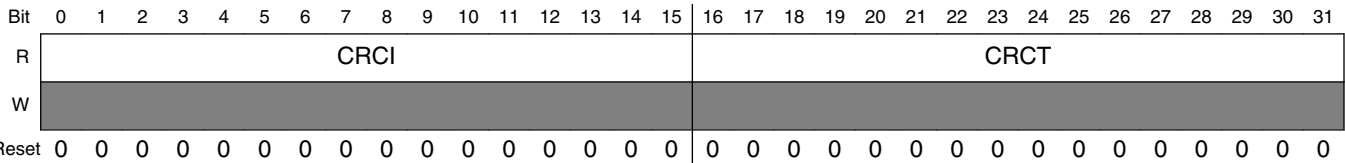
SIPI_CTOR1 field descriptions

Field	Description
0–23 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
24–31 TOR	Timeout value for transmitted requests. Timeout counter runs on the prescaled peripheral clock. Prescaler is defined by SIPI_MCR[PRSCLR]. NOTE: Field can only be written during Initialization mode (SIPI_MCR[INIT] = 1).

44.16.12 SIPI Channel CRC Register 1 (SIPI_CCRC1)

SIPI_CCRCn is a read only which returns the CRC calculated value on the header, address and data bits.

Address: 0h base + 34h offset = 34h



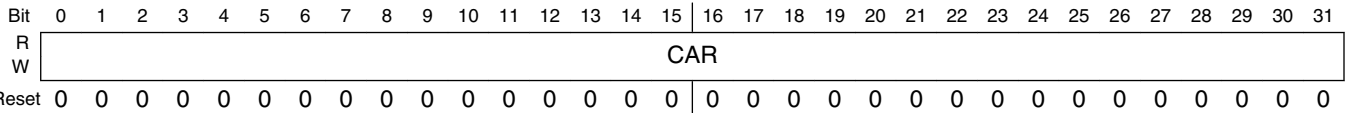
SIPI_CCRC1 field descriptions

Field	Description
0–15 CRCI	Reflects received CRC value at initiator
16–31 CRCT	Reflects received CRC value at target

44.16.13 SIPI Channel Address Register 1 (SIPI_CAR1)

SIPI_CAR1 is the address target for data transmission.

Address: 0h base + 38h offset = 38h



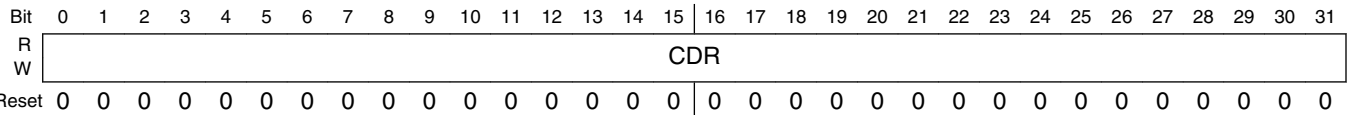
SIPI_CAR1 field descriptions

Field	Description
0–31 CAR	These bits contain the address of the target node.

44.16.14 SIPI Channel Data Register 1 (SIPI_CDR1)

SIPI_CDR1 contains the data to be transmitted, and can be written anytime by software or the DMA. The data can be written in 8, 16 or 32 bit formats.

Address: 0h base + 3Ch offset = 3Ch



SIPI_CDR1 field descriptions

Field	Description
0–31 CDR	Data register bits. Contains the data that will be transmitted, or received.

44.16.15 SIPI Channel Control Register 2 (SIPI_CCR2)

NOTE

Back to back transactions are not supported on a single channel (for example, if the channel busy bit, SIPI_CSR2[CB] = 1, is configured for a channel, it cannot be triggered for another transfer until the data buffers are empty and the channel is free, SIPI_CSR2[CB] = 0).

PRIORITY SCHEDULING: The channel whose SIPI_CAR_n is written first will gain access first irrespective of its priority at the start of the transfer. As more transfers are requested, priority will be resolved as per the priority scheme (for example, channel 0 – channel 1 – channel 2 – channel 3). Scheduling will occur each time the LFAST is not ready to receive data and no channel is pending for data transmission. The reason for the scheduling is that a common CRC module is shared by all channels. If the LFAST is ready to receive data during the entire process, it will happen only in the beginning and later on priority will be resolved as per the priority scheme.

NOTE

This register is only writable in Initialization mode (SIPI_MCR[INIT] = 1), except for command bits (for example, TC, WL, ST, IDT, RRT and WRT). Also, the command bits can be written only when the corresponding Channel busy bit is not asserted (SIPI_CSR n [CB] = 1 (see [SIPI Channel Status Register 2 \(SIPI_CSR2\)](#)). Software will need to poll the respective channel busy bits before attempting to change the command type.

Only one channel at a time can transmit. Therefore, only one channel can remain busy. Maximum time allowed to transmit a full command is nine clock cycles. The Command type of a channel can only be changed when it is no longer busy. Also, to send a command, we need to write to the corresponding SIPI_CAR n (see [SIPI Channel Address Register 2 \(SIPI_CAR2\)](#)).

Address: 0h base + 40h offset = 40h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0								WL		CHEN	ST	IDT	RRT	WRT	DEN
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

SIPI_CCR2 field descriptions

Field	Description
0–14 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
15 TC	Send Trigger Command. A Trigger Command requires that only this bit to be set by software, the trigger does not depend on settings in SIPI_CAR2. 0 Trigger command not sent 1 Trigger command sent

Table continues on the next page...

SIPI_CCR2 field descriptions (continued)

Field	Description
16–23 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
24–25 WL	Word Length Transfer. For Streaming write, WL bits should be written 10. 00 8-bit 01 16-bit 10 32-bit 11 not used
26 CHEN	Channel Enable. If all channel enables are simultaneously written to enable the channels, channel 0 will be given the highest priority for transmission. For channels, the lowest channel number is given highest priority. 0 Channel is disabled 1 Channel is enabled
27 ST	This bit is hard-coded to 0. It can only be written for channel 2 (SIPI_CCR2). This bit can only be written when the corresponding SIPI_CCRn[WRT] = 1. NOTE: Only transfers with channel 2 can write 1 to this bit. All other channels force SIPI_CCRn[ST] = 0. The SIPI_CCR2[WRT] and SIPI_CCR2[ST] bits must be set for streaming. 0 Streaming transfer is disabled 1 Streaming transfer is enabled
28 IDT	ID Read Request Transfer. This request returns the value of the CHIP ID. NOTE: The order of priority for writing to IDT, WRT and RRT must be in this order: 1. IDT 2. RRT 3. WRT 0 ID read request not sent 1 ID read request sent
29 RRT	Read Request Transfer. NOTE: The order of priority for writing to IDT, WRT and RRT must be in this order: 1. IDT 2. RRT 3. WRT 0 Read request will not be sent 1 Read request transfer by the initiator. This bit can not be written if SIPI_CCR2[IDT] = 1.
30 WRT	Write Request Transfer. NOTE: The order of priority for writing to IDT, WRT and RRT must be in this order: 1. IDT 2. RRT 3. WRT

Table continues on the next page...

SIPI_CCR2 field descriptions (continued)

Field	Description
	0 No write request will be sent 1 Write request transfer by the initiator. This bit can not be written if SIPI_CCR2[IDT] = 1 or SIPI_CCR2[RRT] = 1.
31 DEN	DMA Enable. When enabling DMA mode, this bit should be set by software. It will then be cleared by hardware after one major loop has completed. 0 Channel will be used for bus interface access. 1 Channel will be used for DMA access

44.16.16 SIPI Channel Status Register 2 (SIPI_CSR2)

SIPI_CSR2 contains the status bits for the current transfer.

Address: 0h base + 44h offset = 44h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0								RAR	TID			ACKR	CB	0	
W									w1c				w1c			
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

SIPI_CSR2 field descriptions

Field	Description
0–23 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
24 RAR	Read Answer Reception.

Table continues on the next page...

SIPI_CSR2 field descriptions (continued)

Field	Description
	0 Read answer not received 1 Read answer received
25–27 TID	Transaction ID of transmitted frame. Transaction ID is a random number associated with every Tx frame to match the received response/ack with the command.
28 ACKR	Acknowledge Received. 0 Acknowledge not received 1 Acknowledge received
29 CB	Channel Busy. Indicates channel 2 status. 0 Channel 2 free 1 Channel 2 busy
30–31 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

44.16.17 SIPI Channel Interrupt Register 2 (SIPI_CIR2)

The SIPI_CIR2 contains the interrupt enable bits for channel 2.

Address: 0h base + 4Ch offset = 4Ch

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	Reserved															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	Reserved															
W											WAIE	RAIE	TCIE	TOIE	TIDIE	ACKIE
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

SIPI_CIR2 field descriptions

Field	Description
0–25 Reserved	This field is reserved.
26 WAIE	Write Acknowledge Interrupt Enable. 0 Interrupt is disabled 1 Interrupt is enabled
27 RAIE	Read Answer Interrupt Enable. 0 Interrupt is disabled 1 Interrupt is enabled
28 TCIE	Trigger Command Interrupt Enable. 0 Interrupt is disabled 1 Interrupt is enabled
29 TOIE	Timeout Error Interrupt Enabled.

Table continues on the next page...

SIPI_CIR2 field descriptions (continued)

Field	Description
	0 Interrupt is disabled 1 Interrupt is enabled
30 TIDIE	Transaction ID Error Interrupt Enable. 0 Interrupt is disabled 1 Interrupt is enabled
31 ACKIE	Acknowledge Error Interrupt Enable. 0 Interrupt is disabled 1 Interrupt is enabled

44.16.18 SIPI Channel Timeout Register 2 (SIPI_CTOR2)

SIPI_CTOR2 contains the timeout value for Tx requests.

Address: 0h base + 50h offset = 50h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0																							TOR								
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1	

SIPI_CTOR2 field descriptions

Field	Description
0–23 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
24–31 TOR	Timeout counter runs on the prescaled peripheral clock. Prescaler is defined by SIPI_MCR[PRSCLR]. NOTE: Field can only be written during Initialization mode (SIPI_MCR[INIT] = 1).

44.16.19 SIPI Channel CRC Register 2 (SIPI_CCRC2)

SIPI_CCRC n is a read only which returns the CRC calculated value on the header, address and data bits.

Address: 0h base + 54h offset = 54h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	CRCI															CRCT																
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	

SIPI_CCRC2 field descriptions

Field	Description
0–15 CRCI	Reflects received CRC value at initiator
16–31 CRCT	Reflects received CRC value at target

44.16.20 SIPI Channel Address Register 2 (SIPI_CAR2)

SIPI_CAR2 is the address target for data transmission. For streaming operations this register is the start address.

Address: 0h base + 58h offset = 58h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R																																
W																																
	CAR																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	

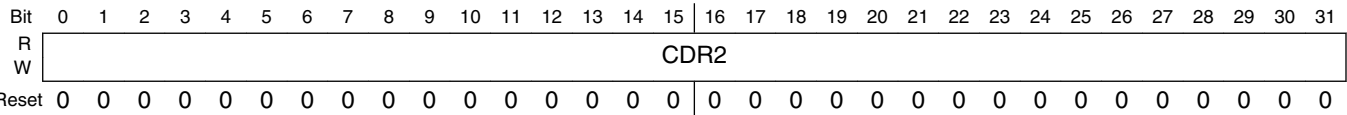
SIPI_CAR2 field descriptions

Field	Description
0–31 CAR	These bits contain the address of the target node.

44.16.21 **SIPI Channel Data Register 2 (SIPI_CDR2_n)**

SIPI_CDR2_n contains the data to be transmitted, and can be written anytime by software or the DMA. The data can be written in 8, 16 or 32 bit formats.

Address: 0h base + 5Ch offset + (4d × i), where i=0d to 7d



SIPI_CDR2_n field descriptions

Field	Description
0–31 CDR2	Data register bits. Contains the data that will be transmitted, or received.

44.16.22 **SIPI Channel Control Register 3 (SIPI_CCR3)**

NOTE

Back to back transactions are not supported on a single channel (for example, if the channel busy bit, SIPI_CSR3[CB] = 1, is configured for a channel, it cannot be triggered for another transfer until the data buffers are empty and the channel is free, SIPI_CSR3[CB] = 0).

PRIORITY SCHEDULING: The channel whose SIPI_CARn is written first will gain access first irrespective of its priority at the start of the transfer. As more transfers are requested, priority will be resolved as per the priority scheme (for example, channel 0 – channel 1 – channel 2 – channel 3). Scheduling will occur each time the LFAST is not ready to receive data and no channel is pending for data transmission. The reason for the scheduling is that a common CRC module is shared by all channels. If the LFAST is ready to receive data during the entire process, it will happen only in the beginning and later on priority will be resolved as per the priority scheme.

NOTE

This register is only writable in Initialization mode (SIPI_MCR[INIT] = 1), except for command bits (for example, TC, WL, ST, IDT, RRT and WRT). Also, the command bits can be written only when the corresponding Channel busy bit is not asserted (SIPI_CSR n [CB] = 1 (see [SIPI Channel Status Register 3 \(SIPI_CSR3\)](#)). Software will need to poll the respective channel busy bits before attempting to change the command type.

Only one channel at a time can transmit. Therefore, only one channel can remain busy. Maximum time allowed to transmit a full command is nine clock cycles. The Command type of a channel can only be changed when it is no longer busy. Also, to send a command, we need to write to the corresponding SIPI_CAR n (see [SIPI Channel Address Register 3 \(SIPI_CAR3\)](#)).

Address: 0h base + 7Ch offset = 7Ch

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0								WL		CHEN	ST	IDT	RRT	WRT	DEN
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

SIPI_CCR3 field descriptions

Field	Description
0–14 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
15 TC	Send Trigger Command. A Trigger Command requires that only this bit to be set by software, the trigger does not depend on settings in SIPI_CAR3. 0 Trigger command not sent 1 Trigger command sent

Table continues on the next page...

SIPI_CCR3 field descriptions (continued)

Field	Description
16–23 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
24–25 WL	Word Length Transfer. For Streaming write WL bits should be written 10. 00 8-bit 01 16-bit 10 32-bit 11 not used
26 CHEN	Channel Enable. If all channel enables are simultaneously written to enable the channels, channel 0 will be given the highest priority for transmission. For channels, the lowest channel number is given highest priority. 0 Channel is disabled 1 Channel is enabled
27 ST	Streaming Transfer. This bit is hard-coded to 0. It can only be written for channel 2 (SIPI_CCR2). This bit can only be written when the corresponding SIPI_CCR _n [WRT] = 1. NOTE: Only transfers with channel 2 can write 1 to this bit. All other channels force SIPI_CCR _n [ST] = 0. The SIPI_CCR2[WRT] and SIPI_CCR2[ST] bits must be set for streaming. 0 Streaming transfer is disabled 1 Streaming transfer is enabled
28 IDT	ID Read Request Transfer. This request returns the value of the CHIP ID. NOTE: The order of priority for writing to IDT, WRT and RRT must be in this order: 1. IDT 2. RRT 3. WRT 0 ID read request not sent 1 ID read request sent
29 RRT	Read Request Transfer. NOTE: The order of priority for writing to IDT, WRT and RRT must be in this order: 1. IDT 2. RRT 3. WRT 0 Read request will not be sent 1 Read request transfer by the initiator. This bit can not be written if SIPI_CCR3[IDT] = 1.
30 WRT	Write Request Transfer. NOTE: The order of priority for writing to IDT, WRT and RRT must be in this order: 1. IDT 2. RRT

Table continues on the next page...

SIPI_CCR3 field descriptions (continued)

Field	Description
	3. WRT
	0 No write request will be sent 1 Write request transfer by the initiator. This bit can not be written if SIPI_CCR3[IDT] = 1 or SIPI_CCR3[RRT] = 1.
31 DEN	DMA Enable. When enabling DMA mode, this bit should be set by software. It will then be cleared by hardware after one major loop has completed. 0 Channel will be used for bus interface access. 1 Channel will be used for DMA access

44.16.23 SIPI Channel Status Register 3 (SIPI_CSR3)

SIPI_CSR3 contains the status bits for the current transfer.

Address: 0h base + 80h offset = 80h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0								RAR	TID				ACKR	CB	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

SIPI_CSR3 field descriptions

Field	Description
0–23 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
24 RAR	Read Answer Reception. 0 Read answer not received 1 Read answer received
25–27 TID	Transaction ID of transmitted frame. Transaction ID is a random number associated with every Tx frame to match the received response/ack with the command.
28 ACKR	Acknowledge Received. 0 Acknowledge not received 1 Acknowledge received
29 CB	Channel Busy. Indicates channel 3 status. 0 Channel 3 free 1 Channel 3 busy
30–31 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

44.16.24 SIPI Channel Interrupt Register 3 (SIPI_CIR3)

SIPI_CIR3 contains the interrupt enable bits for channel 3.

Address: 0h base + 88h offset = 88h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	Reserved															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	Reserved															
W											WAIE	RAIE	TCIE	TOIE	TIDIE	ACKIE
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

SIPI_CIR3 field descriptions

Field	Description
0–25 Reserved	This field is reserved.
26 WAIE	Write Acknowledge Interrupt Enable. 0 Interrupt is disabled 1 Interrupt is enabled
27 RAIE	Read Answer Interrupt Enable. 0 Interrupt is disabled 1 Interrupt is enabled
28 TCIE	Trigger Command Interrupt Enable. 0 Interrupt is disabled 1 Interrupt is enabled
29 TOIE	Timeout Error Interrupt Enabled.

Table continues on the next page...

SIPI_CIR3 field descriptions (continued)

Field	Description
	0 Interrupt is disabled 1 Interrupt is enabled
30 TIDIE	Transaction ID Error Interrupt Enable. 0 Interrupt is disabled 1 Interrupt is enabled
31 ACKIE	Acknowledge Error Interrupt Enable. 0 Interrupt is disabled 1 Interrupt is enabled

44.16.25 SIPI Channel Timeout Register 3 (SIPI_CTOR3)

SIPI_CTOR3 contains the timeout value for Tx requests.

Address: 0h base + 8Ch offset = 8Ch

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0																							TOR								
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1	

SIPI_CTOR3 field descriptions

Field	Description
0–23 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
24–31 TOR	Timeout value for transmitted requests. Timeout counter runs on the prescaled peripheral clock. Prescaler is defined by SIPI_MCR[PRSCLR]. NOTE: Field can only be written during Initialization mode (SIPI_MCR[INIT] = 1).

44.16.26 SIPI Channel CRC Register 3 (SIPI_CCRC3)

SIPI_CCRC n is a read only which returns the CRC calculated value on the header, address and data bits.

Address: 0h base + 90h offset = 90h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	CRCI															CRCT																
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	

SIPI_CCRC3 field descriptions

Field	Description
0–15 CRCI	Reflects received CRC value at initiator
16–31 CRCT	Reflects received CRC value at target

44.16.27 SIPI Channel Address Register 3 (SIPI_CAR3)

SIPI_CAR3 is the address target for data transmission.

Address: 0h base + 94h offset = 94h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	CAR																															
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	

SIPI_CAR3 field descriptions

Field	Description
0–31 CAR	These bits contain the address of the target node.

44.16.28 SIPI Channel Data Register 3 (SIPI_CDR3)

SIPI_CDR3 contains the data to be transmitted, and can be written anytime by software or the DMA. The data can be written in 8, 16 or 32 bit formats.

Address: 0h base + 98h offset = 98h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	CDR																															
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	

SIPI_CDR3 field descriptions

Field	Description
0–31 CDR	Data register bits. Contains the data that will be transmitted, or received.

44.16.29 SIPI Module Configuration Register (SIPI_MCR)

The SIPI_MCR is a global 32-bit configuration register.

Address: 0h base + 9Ch offset = 9Ch

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	FRZ	Reserved	HALT	0		PRSCLR										
W																
Reset	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	AID	0				CRCIE	MCRIE	0				CHNSB	TEN	INIT	MOEN	SR
W																
Reset	0*	0*	0	0	0	0*	0*	0	0	0	0	0*	0*	0*	0	0

* Notes:

- INIT field: Can only be written when SIPI_MCR[MOEN] = 1.
- TEN field: Can only be written when SIPI_MCR[MOEN] = 1.
- CHNSB field: Can be written only once after reset.
- MCRIE field: Can only be written when SIPI_MCR[MOEN] = 1.
- CRCIE field: Can only be written when SIPI_MCR[MOEN] = 1.
- AID field: Can be written in initialization mode only (SIPI_MCR[INIT] = 1).

SIPI_MCR field descriptions

Field	Description
0 FRZ	Freeze Enable The FRZ bit specifies the SIPI behavior when Debug mode is requested at the MCU level. When FRZ is asserted, the SIPI is enabled to enter Freeze mode. Negation of this bit field causes SIPI to exit Freeze mode. NOTE: Can only be written when SIPI_MCR[MOEN] = 1. 0 Not enabled to enter Freeze mode 1 Enabled to enter Freeze mode
1 Reserved	This field is reserved. Important: Always write the default value to this field.
2 HALT	Halt Mode Enable Assertion of this bit puts SIPI into Freeze mode. No Rx or Tx is performed in the SIPI until this bit is cleared. If this bit is enabled in during Tx or Rx communications, the current activity will finish, then the SIPI will enter Freeze mode. NOTE: Can only be written when SIPI_MCR[MOEN] = 1. 0 No Freeze mode request 1 Enters Freeze mode if FRZ bit is asserted.
3–4 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
5–15 PRSCLR	Timeout counter prescaler The timeout counter runs on the prescaled system clock. Default value is 64 (040h). The allowed programmable values are (all other values are ignored): 040h 64 (default) 080h 128 100h 256 200h 512 400h 1024 NOTE: This field should be programmed by software during initialization mode, SIPI_MCR[INIT] = 1. Writes during other times are ignored. NOTE: Writes to SIPI_MCR[PRSCLR] can only be accomplished with 16-bit or 32-bit writes.
16–17 AID	Address Increment/Decrement These bits define the type of address change at the target node. NOTE: This bits should be programmed by software in initialization mode, SIPI_MCR[INIT] = 1. Writes during other times are ignored. 00 no change. address stays same 01 address increments by 4 10 address decrements by 4 11 not used
18–20 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

Table continues on the next page...

SIPI_MCR field descriptions (continued)

Field	Description
21 CRCIE	CRC Error Interrupt Enable 0 Interrupt is disabled 1 Interrupt is enabled
22 MCRIE	Max Count Reached Interrupt Enable 0 Interrupt is disabled 1 Interrupt is enabled
23–26 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
27 CHNSB	Channel coding select bit. 0 Code Table II (see Table 44-2) 1 Code Table I (see Table 44-2)
28 TEN	Target Enable Setting this bit enables the target functionality at SIPI. This bit can be read or written anytime. This bit is automatically negated by hardware when target detects an error in "streaming without ACK" mode. This bit has to be enabled for the transmission operations also.
29 INIT	Initialization Mode Setting this bit puts the module in initialization mode. This bit should be cleared by software. Most register bits are configured using this bit. The SIPI_MCR[MOEN] bit needs to be set first, and then the INIT bit can be set and both bits can't be enabled together. 0 Normal Mode 1 Initialization Mode
30 MOEN	Module Enable This bit should be set or cleared by software. When this bit is negated, future SIPI Tx transfers are disabled.
31 SR	Soft Reset Setting this bit clears all status and error registers, and FSMs are moved to idle state. This bit is automatically cleared by hardware once the reset operation is complete.

44.16.30 SIPI Status Register (SIPI_SR)

The SIPI_SR is the global status register of SIPI.

Address: 0h base + A0h offset = A0h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	FRZACK	Reserved	Reserved													
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	Reserved					GCRCE	MCR	Reserved	TE				STATE			
W						w1c	w1c		w1c							
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

SIPI_SR field descriptions

Field	Description
0 FRZACK	Freeze Mode Acknowledge. This read-only bit indicates that SIPI is in Freeze mode. The Freeze mode request cannot be granted until current transmission or reception processes have finished. The software can poll the FRZACK bit to find when the SIPI has actually entered Freeze mode. If Freeze mode is requested while SIPI is in any of the low power modes, then the FRZACK bit will only be set when the low power mode is exited.

Table continues on the next page...

SIPI_SR field descriptions (continued)

Field	Description
	0 SIPI not in Freeze mode 1 SIPI in Freeze mode
1 Reserved	This field is reserved.
2–20 Reserved	This field is reserved.
21 GCRCE	Global CRC Error Bit. 0 No CRC error 1 CRC error occurred
22 MCR	Maximum Count Reached. This bit will be set whenever SIPI_ACR[ADCNT] = SIPI_MAXCR[MXCNT]. An interrupt will be generated when this bit is set only if SIPI_MCR[MCRIE] = 1. This it should be cleared by software.
23 Reserved	This field is reserved.
24–27 TE	Trigger Event on Respective Channels. This field enables interrupts for target nodes. xxx1 TE0 = 1 - Channel 0 trigger event xx1x TE1 = 1 - Channel 1 trigger event x1xx TE2 = 1 - Channel 2 trigger event 1xxx TE3 = 1 - Channel 3 trigger event
28–31 STATE	These bits reflect the transmit state machine status. They can be polled for determination of state machine status. 0000 IDLE 0001 HEADER_AND_ADDRESS_FIELD 0010 HEADER_AND_DATA_FIELD 0011 HEADER_AND_CRC_FIELD 0100 ADDRESS_AND_CRC_FIELD 0101 ADDRESS_AND_DATA_FIELD 0110 DATA_AND_CRC_FIELD 0111 DATA_FIELD

44.16.31 SIPI Max Count Register (SIPI_MAXCR)

SIPI_MAXCR contains the maximum address count value at target node. It is programmed via direct write request through the initiator. This register can be read or written by software anytime.

Address: 0h base + A4h offset = A4h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	MXCNT															
W																
Reset	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	MXCNT														Reserved	
W																
Reset	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0	0

SIPI_MAXCR field descriptions

Field	Description
0–29 MXCNT	This field contains the maximum address count value at the target node. It should be programmed via direct write request through the initiator.
30–31 Reserved	This field is reserved.

44.16.32 SIPI Address Reload Register (SIPI_ARR)

The SIPI_ARR contains the reload value for the address counter at the target node. It should be configured by direct write request from the initiator.

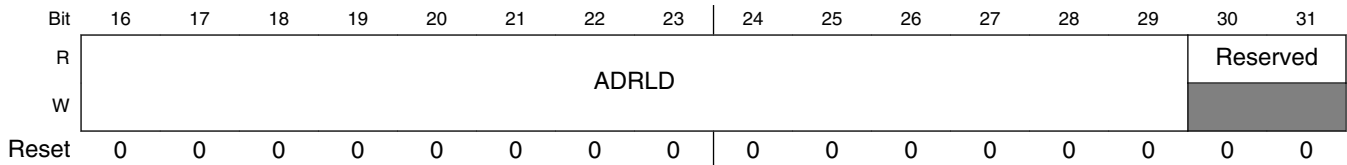
NOTE

This register is writeable only when SIPI_MCR[INIT] = 1.

Address: 0h base + A8h offset = A8h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	ADRLD															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Memory map and register definition



SIPI_ARR field descriptions

Field	Description
0–29 ADRLD	ADRLD contains the reload value for the address counter at the target node. It should be configured by direct write request from the initiator.
30–31 Reserved	This field is reserved.

44.16.33 SIPI Address Count Register (SIPI_ACR)

This register reflects the count value of address counter at target node. It should be configured by direct write request from initiator. This register can be read/written by software anytime.

NOTE

SIPI_ARR, SIPI_ACR and SIPI_MAXCR will be configured by direct write operation through the initiator.

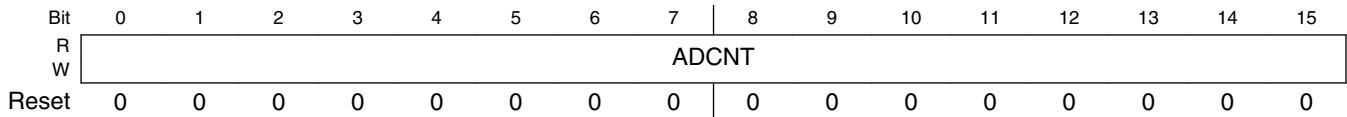
NOTE

When a streaming write command is received, the target will start the write operation from the address present in SIPI_ACR (address count register).

NOTE

After each 32-bit write has completed, SIPI_ACR[ADCNT] is compared against SIPI_MAXCR[MXCNT]. If they are equal, SIPI_ACR[ADCNT] is loaded with the value stored in SIPI_ARR[ADRLD]. If they are not equal, the SIPI_ACR[ADCNT] will increment by 4, decrement by 4 or remain the same (depending on configuration). In both cases, the SIPI_ACR[ADCNT] will contain the address at which the next streaming write will occur.

Address: 0h base + ACh offset = ACh



Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	ADCNT														Reserved	
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

SIPI_ACR field descriptions

Field	Description
0–29 ADCNT	Reflects the count value of address counter at target node. It should be configured by direct write request from the initiator. This register can be read/written by software anytime. At the end of the streaming operation, SIPI_ACR will point to the next address to be written.
30–31 Reserved	This field is reserved.

44.16.34 SIPI Error Register (SIPI_ERR)

This register contains the error bits for the last transaction(s) for all the channels. Communication errors are generated only for out of range address accesses, areas that are read only and where accesses do not lead to generation of transfer errors. Also, writing to read only registers will not generate errors.

Address: 0h base + B0h offset = B0h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0					TOE3	TIDE3	ACE3	0					TOE2	TIDE2	ACE2
W						w1c	w1c	w1c						w1c	w1c	w1c
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0					TOE1	TIDE1	ACE1	0					TOE0	TIDE0	ACE0
W						w1c	w1c	w1c						w1c	w1c	w1c
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

SIPI_ERR field descriptions

Field	Description
0–4 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
5 TOE3	Timeout Error for Channel 3. 0 Timeout error didn't occur 1 Timeout error occurred
6 TIDE3	Transaction ID Error for Channel 3. 0 Received transaction ID matched with the stored ID 1 Received transaction ID didn't match with the stored ID
7 ACKE3	Acknowledge Error for Channel 3. 0 Acknowledge received is correct. 1 Acknowledge received is not correct.
8–12 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
13 TOE2	Timeout Error for Channel 2. 0 Timeout error didn't occur 1 Timeout error occurred
14 TIDE2	Transaction ID Error for Channel 2. 0 Received transaction ID matched with the stored ID 1 Received transaction ID didn't match with the stored ID
15 ACKE2	Acknowledge Error for Channel 2. 0 Acknowledge received is correct 1 Acknowledge received is not correct
16–20 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
21 TOE1	Timeout Error for Channel 1. 0 Timeout error didn't occur 1 Timeout error occurred
22 TIDE1	Transaction ID Error for Channel 1. 0 Received transaction ID matched with the stored ID 1 Received transaction ID didn't match with the stored ID
23 ACKE1	Acknowledge Error for Channel 1. 0 Acknowledge received is correct 1 Acknowledge received is not correct
24–28 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
29 TOE0	Timeout Error for Channel 0. 0 Timeout error didn't occur 1 Timeout error occurred

Table continues on the next page...

SIPI_ERR field descriptions (continued)

Field	Description
30 TIDE0	Transaction ID Error for Channel 0. 0 Received transaction ID matched with the stored ID 1 Received transaction ID didn't match with the stored ID
31 ACKE0	Acknowledge Error for Channel 0. 0 Acknowledge received is correct. 1 Acknowledge received is not correct.

Chapter 45

LVDS Fast Asynchronous Serial Transmission (LFAST) – Interprocessor Communications

45.1 Chip-specific LFAST information

This chip has one instance of the LFAST module.

45.2 Introduction

This chapter describes the specifications of the LFAST module, which implements the LVDS Fast Asynchronous Serial Transmission (LFAST) module. LFAST is used in dual mode (software configurable master/slave operation) for interprocessor communications.

45.3 Block diagram

The following figure depicts LFAST interaction with other modules on the device.

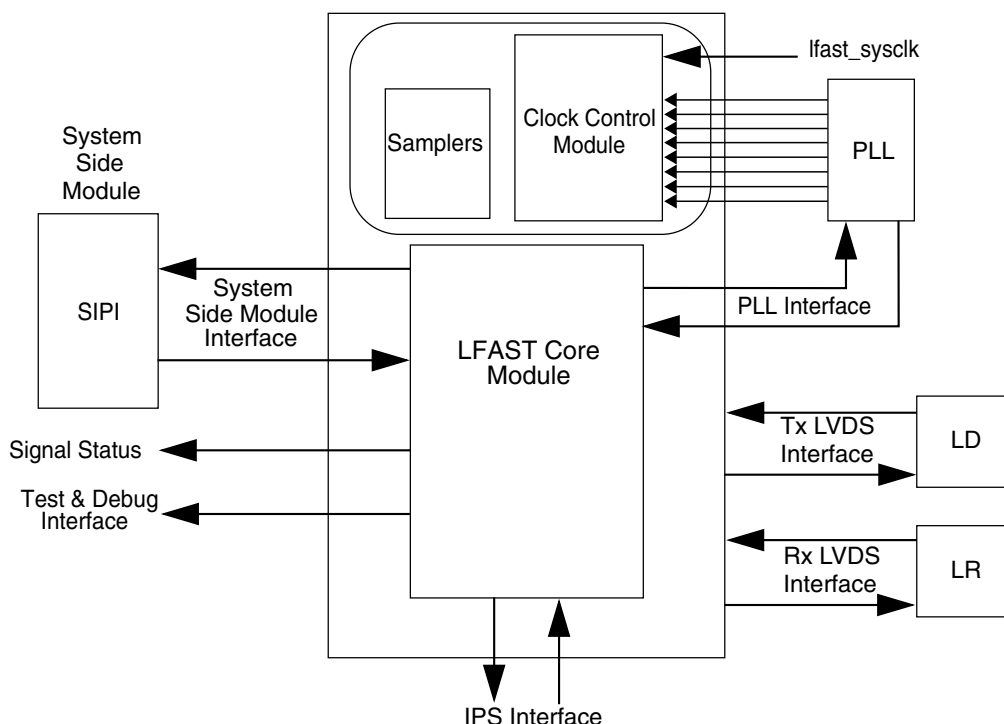


Figure 45-1. LFAST block diagram

45.4 External signals

LFAST is a five pin interface with the following signals :

- lfast_sysclk
 - Reference clock of the LFAST master and slave
- txdatap/txdatan
 - Differential transmit (Tx) interface pair
- rxdatap/rxdatan
 - Differential receive (Rx) interface pair

LFAST interface is an asynchronous high speed LVDS interface.

45.4.1 LFAST operating data rates

The change of data rate is controlled by the LFAST master by issuing appropriate Interface Control Logical Channel (ICLC) packets to the LFAST slave. Henceforth, the data rate 6.5 Mbps/5 Mbps ($\text{lfast\_sysclk} \div 4$ or $\text{lfast\_sysclk} \div 2$) is referred to as low data rate, and the high data rate is in the Data Sheet.

45.5 LFAST frame structure

A LFAST frame is made up of three fields:

- Sync pattern
- Header
- Payload

Sync pattern and header fields are of fixed length.

Sync pattern is used to synchronize incoming data stream in LFAST module.

Header field of the frame distinguishes various types of data and control transferred and also contains information about the length of the payload. The payload field is the actual data that is transferred across the channel. See [Figure 45-2](#) for details of the LFAST frame.

LFAST frame structure

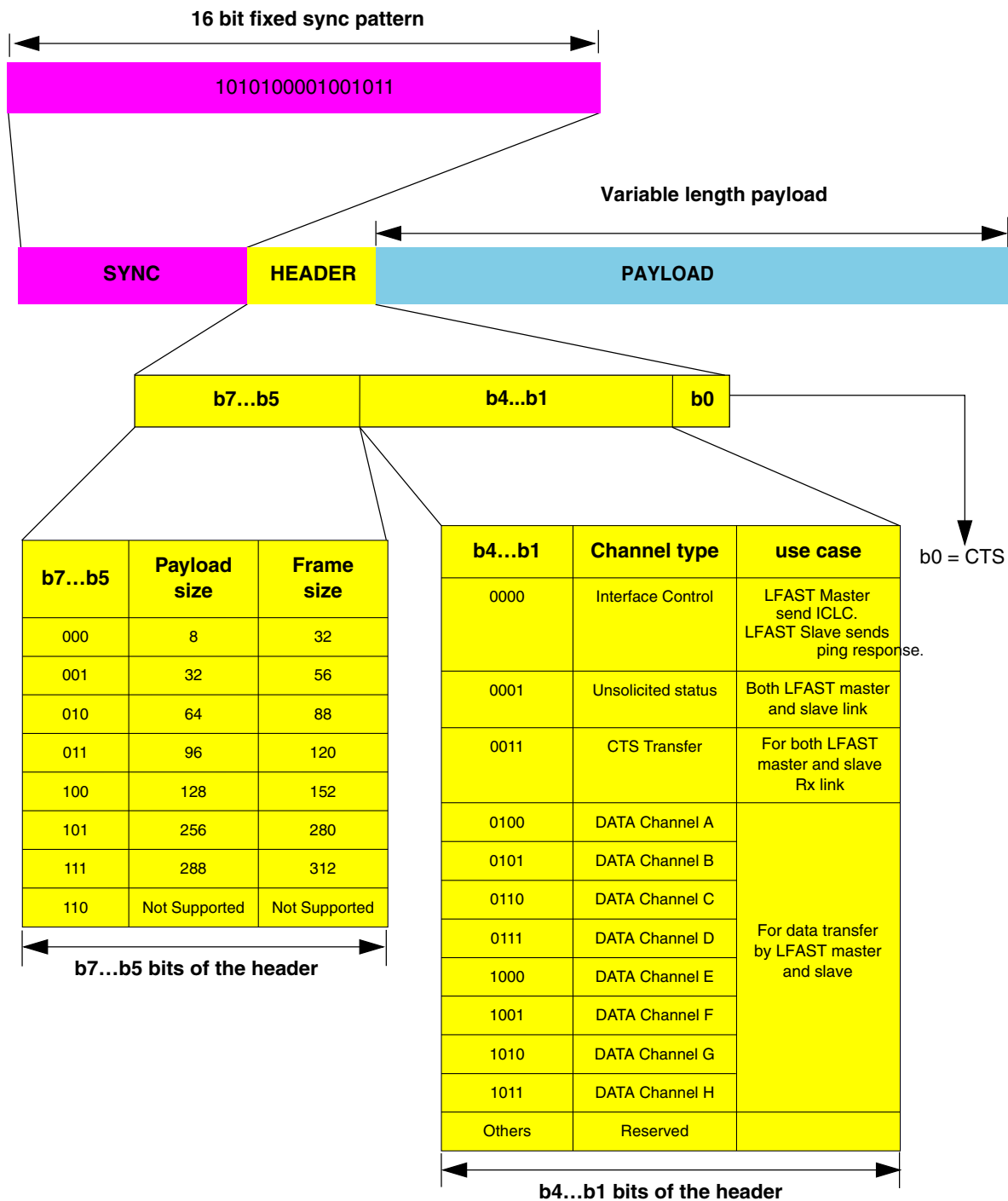


Figure 45-2. LFAST frame structure

The same protocol is used on both transmit and receive interfaces for communications of data, control and status information. A synchronization pattern (16 bits) and header (8 bits) are present in every frame.

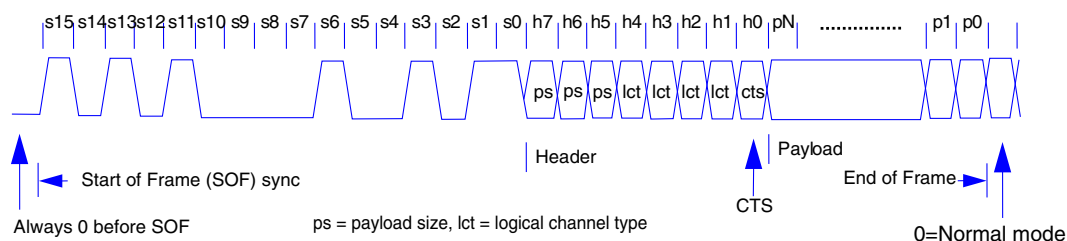


Figure 45-3. Serial frame structure

The frame structure is shown in [Figure 45-3](#) and consists of the following:

- **16-bit synchronization pattern:** Used for clock synchronization and pattern recognition. The frame synchronization code at the start of every frame is a unique reserved word that is used to identify whether the data received is the start of the frame and for synchronization (clock phase extraction) with the stream data. A synchronous sequence is used to give a high quality auto correlation. The LFAST 16-bit synchronization sequence = 1010_1000_0100_1011b = A84Bh.
- **Header - 3 MSB (b7 - b5):** Defines the payload size as shown in [Table 45-1](#) :

Table 45-1. Header payload sizes

b7-b5(bin)	Payload Size	Frame Size
000	8	32
001	32	56
010	64	88
011	96	120
100	128	152
101	256	280
110	—	—
111	288	312

- **Header - (b4 - b1):** Defines the logical channel types, which indicate the type of payload that the frame carries. How the payload field of a frame is used for any other logical channel type is system side module specific except in the case of the interface control logical channel type and clear to send (CTS) frame.
- **Header - (b0):** CTS on the both LFAST master and slave devices.
- **Payload:** Content dependent upon frame type.
- **Bit after frame:** This bit determines entry into Sleep mode (1 = Sleep mode, 0 = normal mode)

45.6 Features

- Supports dual mode (register configurable Master/Slave).
- Supports asynchronous data transfer up to the maximum data rate shown in the product Data Sheet.
- Transmits and receives data, CTS, ICLC and unsolicited frames.
- Receives ICLC frames
- Provision of five interrupts for Tx and Rx channels.
- Supports processor controlled transfer of ICLC frame with 8-bit payload size to implement the data rate changes and test modes.
- Supports LFAST defined CTS controlled data transfer. No dependency between data transmission and reception unless CTS mode is enabled.
- Supports flow control using sliding window protocol.
- Provides configurable frame length for data frame with variable payload sizes of 32, 64, 96, 128, 256 or 288 bits.
- Provides transmit of data frame length with 96 bits of payload size and reception of data frame with 128 bits of payload size.
- Provides configurable frame length for unsolicited frame with variable payload sizes of 8, 32, 64, 96, 128, 256 or 288 bits.
- Supports PLL configuration (for example, feedback loop divider, etc.) through registers.
- Supports LVDS configuration through registers.
- Supports multiple loopback modes for checking the physical interface.
- Supports automatic ping response generation in slave mode.
- Supports for detection of unsupported channel number and unsupported payload size.

45.7 Memory map and register definition

All registers are 32 bits wide.

NOTE

Read/Write accesses to all unimplemented registers and write accesses to Read only register will return a Transfer Error.

LFAST memory map

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
0	LFAST Mode Configuration Register (LFAST_MCR)	32	R/W	See section	45.7.1/2321
4	LFAST Speed Control Register (LFAST_SCR)	32	R/W	0001_0000h	45.7.2/2323
8	LFAST Correlator Control Register (LFAST_COCCR)	32	R/W	0000_000Eh	45.7.3/2324
C	LFAST Test Mode Control Register (LFAST_TMCR)	32	R/W	0000_0000h	45.7.4/2326
10	LFAST Auto Loopback Control Register (LFAST_ALCR)	32	R/W	0000_0000h	45.7.5/2327
14	LFAST Rate Change Delay Control Register (LFAST_RCDCCR)	32	R/W	000F_0000h	45.7.6/2328
18	LFAST Wakeup Delay Control Register (LFAST_SLCR)	32	R/W	1201_5F02h	45.7.7/2328
1C	LFAST ICLC Control Register (LFAST_ICR)	32	R/W	0000_0000h	45.7.8/2330
20	LFAST Ping Control Register (LFAST_PICR)	32	R/W	0000_80CAh	45.7.9/2331
2C	LFAST Rx FIFO CTS Control Register (LFAST_RFCR)	32	R/W	000F_0009h	45.7.10/2332
30	LFAST Tx Interrupt Enable Register (LFAST_TIER)	32	R/W	0000_0000h	45.7.11/2332
34	LFAST Rx Interrupt Enable Register (LFAST_RIER)	32	R/W	0000_0000h	45.7.12/2333
38	LFAST Rx ICLC Interrupt Enable Register (LFAST_RIIER)	32	R/W	0000_0000h	45.7.13/2335
3C	LFAST PLL Control Register (LFAST_PLLCR)	32	R/W	0000_005Ch	45.7.14/2337
40	LFAST LVDS Control Register (LFAST_LCR)	32	R/W	See section	45.7.15/2339
44	LFAST Unsolicited Tx Control Register (LFAST_UNSTCR)	32	R/W	0000_0000h	45.7.16/2342
48	LFAST Unsolicited Tx Data Registers (LFAST_UNSTDR0)	32	R/W	0000_0000h	45.7.17/2342
4C	LFAST Unsolicited Tx Data Registers (LFAST_UNSTDR1)	32	R/W	0000_0000h	45.7.17/2342
50	LFAST Unsolicited Tx Data Registers (LFAST_UNSTDR2)	32	R/W	0000_0000h	45.7.17/2342
54	LFAST Unsolicited Tx Data Registers (LFAST_UNSTDR3)	32	R/W	0000_0000h	45.7.17/2342

Table continues on the next page...

LFAST memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
58	LFAST Unsolicited Tx Data Registers (LFAST_UNSTDR4)	32	R/W	0000_0000h	45.7.17/ 2342
5C	LFAST Unsolicited Tx Data Registers (LFAST_UNSTDR5)	32	R/W	0000_0000h	45.7.17/ 2342
60	LFAST Unsolicited Tx Data Registers (LFAST_UNSTDR6)	32	R/W	0000_0000h	45.7.17/ 2342
64	LFAST Unsolicited Tx Data Registers (LFAST_UNSTDR7)	32	R/W	0000_0000h	45.7.17/ 2342
68	LFAST Unsolicited Tx Data Registers (LFAST_UNSTDR8)	32	R/W	0000_0000h	45.7.17/ 2342
80	LFAST Global Status Register (LFAST_GSR)	32	R	See section	45.7.18/ 2343
84	LFAST Ping Status Register (LFAST_PISR)	32	R	0000_0000h	45.7.19/ 2345
94	LFAST Data Frame Status Register (LFAST_DFSR)	32	R	0000_0000h	45.7.20/ 2345
98	LFAST Tx Interrupt Status Register (LFAST_TISR)	32	R/W	0000_0000h	45.7.21/ 2346
9C	LFAST Rx Interrupt Status Register (LFAST_RISR)	32	R/W	0000_0000h	45.7.22/ 2347
A0	LFAST Rx ICLC Interrupt Status Register (LFAST_RIISR)	32	w1c	0000_0000h	45.7.23/ 2349
A4	LFAST PLL and LVDS Status Register (LFAST_PLLLSR)	32	R	0002_0003h	45.7.24/ 2351
A8	LFAST Unsolicited Rx Status Register (LFAST_UNSRSR)	32	R/W	0000_0000h	45.7.25/ 2352
AC	LFAST Unsolicited Rx Data Register (LFAST_UNSRDR0)	32	R	0000_0000h	45.7.26/ 2353
B0	LFAST Unsolicited Rx Data Register (LFAST_UNSRDR1)	32	R	0000_0000h	45.7.26/ 2353
B4	LFAST Unsolicited Rx Data Register (LFAST_UNSRDR2)	32	R	0000_0000h	45.7.26/ 2353
B8	LFAST Unsolicited Rx Data Register (LFAST_UNSRDR3)	32	R	0000_0000h	45.7.26/ 2353
BC	LFAST Unsolicited Rx Data Register (LFAST_UNSRDR4)	32	R	0000_0000h	45.7.26/ 2353
C0	LFAST Unsolicited Rx Data Register (LFAST_UNSRDR5)	32	R	0000_0000h	45.7.26/ 2353
C4	LFAST Unsolicited Rx Data Register (LFAST_UNSRDR6)	32	R	0000_0000h	45.7.26/ 2353
C8	LFAST Unsolicited Rx Data Register (LFAST_UNSRDR7)	32	R	0000_0000h	45.7.26/ 2353
CC	LFAST Unsolicited Rx Data Register (LFAST_UNSRDR8)	32	R	0000_0000h	45.7.26/ 2353

45.7.1 LFAST Mode Configuration Register (LFAST_MCR)

Address: 0h base + 0h offset = 0h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	MSEN	0						IPGDBG	0						LSSEL	
W																
Reset	0*	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	DRFEN	RXEN	TXEN	0								TXARBD	CTSEN	0	DRFRST	DATAEN
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	1	0*	0	0*	0

* Notes:

- DRFRST field: Set by user software and then cleared by system hardware.
- CTSEN field: Only writable when MCR[DRFEN] = 0.
- MSEN field: Writable only once after the asynchronous reset.

LFAST_MCR field descriptions

Field	Description
0 MSEN	LFAST Master or Slave mode Enable. This bit selects either the LFAST master or slave functionality. 0 Enable the modules LFAST Slave functionality only. 1 Enable the modules LFAST Master functionality only.
1–6 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
7 IPGDBG	Control bit to enable support for IPG Debug mode. This mode is indicated by assertion of IPG debug mode signal. 0 IPG debug mode enable signal will be ignored. 1 IPG debug mode enable signal will not be ignored.
8–14 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
15 LSSEL	Selects the fraction of sysclk in Low Speed Select mode (see Slow speed clock for details). 0 Low Speed Mode in which the lfast_sysclk input is used to generate 4 phases of lfast_sysclk/2. 1 Low Speed Mode in which the lfast_sysclk input is used to generate 4 phases of lfast_sysclk/4.
16 DRFEN	LFAST Enable. This bit enables/disables the reception and transfer of LFAST device. 0 LFAST is immediately disabled. All current/pending requests are terminated and the Tx and Rx data FIFOs are flushed. If this bit is cleared in the middle of a transmit/receive operation, then that operation is terminated immediately and nothing is transmitted/received further. All the programmable

Table continues on the next page...

LFAST_MCR field descriptions (continued)

Field	Description
	registers retain their values and status registers are cleared to their reset values. Registers read/write operations can be performed through the IPS Bus. 1 LFAST is Enabled.
17 RXEN	LFAST Receiver Enable. This bit controls the reception of the frames and decoding on the LFAST device. This bit also disables the Rx LVDS Line Receiver (LR). 0 Receiver Interface is disabled. If this bit is cleared during a data transfer, the current frame is received and then the Rx block is disabled. After the Rx block is disabled, all new frames from LFAST peer device are ignored. System Side Module Rx interface isn't disabled by this bit. 1 Receiver Interface is Enabled.
18 TXEN	LFAST Transmitter Enable. This bit controls the transmission of frames from the LFAST device and disables the Tx LVDS LD. This bit can also be modified by LFAST slave H/W on reception of an ICLC command frame. This field can only be written in dual mode (LFAST_GSR[DUALMD] = 1). 0 LFAST transmitter Interface is disabled. No new request is accepted but ongoing request is served. 1 LFAST transmitter Interface is enabled. New requests are accepted.
19–26 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
27 TXARBD	Tx Arbiter Disable. This bit enables/disables the Tx block arbiter. Current frame transfer is completed, but new frame requests are ignored. 0 Enable Tx arbiter and framer. When enabled it takes all the frame request and services based on priority. 1 Disable Tx arbiter and framer. All frame requests are ignored.
28 CTSEN	CTS Enable. This bit defines the Push-Pull mode of the LFAST devices receiver. This bit is used to enable/disable CTS mode of the Tx block. This bit is only writable when MCR[DRFEN] = 0. 0 CTS mode is disabled. Indicates that the device is in Push mode. The CTS bit of frames transmitted is 1. The CTS bit doesn't represent the status of Rx FIFO. 1 CTS mode is enabled. The CTS bit of all transmit frames is set when the Rx FIFO empty space is on or above higher threshold, and cleared when the Rx FIFO empty space is on or below lower threshold.
29 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
30 DRFRST	LFAST Soft Reset. This bit is automatically cleared after Reset.. 0 No Soft Reset 1 Soft Reset to LFAST is asserted. When set it causes a reset of the LFAST module; all the registers will be reset to their default values and all the FIFOs will be flushed.
31 DATAEN	DATA Frame Enable. This bit enables/disables the transmission and reception of data frames between the LFAST master and slave devices. 0 Data frame transmission and reception is disabled. Tx data frame requests are ignored by the transmitter. Frame with LCT of data frame is ignored by the receiver. 1 Data frame transmission and reception is enabled. Tx data frame requests are serviced by the transmitter. Frame with LCT of data frame is received and placed into the Rx data FIFO.

45.7.2 LFAST Speed Control Register (LFAST_SCR)

The SCR is used to configure the Rx and Tx data rate of the LFAST.

Address: 0h base + 4h offset = 4h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1
Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0								0							
W								RDR								TDR
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

LFAST_SCR field descriptions

Field	Description
0–14 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
15 DRMD	Data Rate Controller mode. Defines the mode setting for LFAST slave device by S/W or LFAST master. 0 S/W controls the Data Rate controller mode. In LFAST Slave the ICLC frames for rate change have no affect on the Data rate. 1 In LFAST Slave the reception of ICLC frame for rate change sets appropriate speed mode. In LFAST Master the SCR[DRMD] should be 0.
16–22 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
23 RDR	Receiver Data Rate. This bit defines the receiver data rate. For LFAST Master: - S/W should program this bit only after transmission of an ICLC frame changing the speed mode of the slaves Tx interface. For LFAST Slave: - When SCR[DRMD] = 1, the H/W programs this bit on reception of ICLC frame for changing speed mode of Slaves Rx interface. - The S/W can program this bit when SCR[DRMD] = 0. - This bit is cleared on MCR[DRFEN] negation. 0 Data rate of Rx block is low speed. 1 Data rate of Rx block is high speed.

Table continues on the next page...

LFAST_SCR field descriptions (continued)

Field	Description
24–30 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
31 TDR	Transmit Data Rate. This bit defines the transmitter data rate. For LFAST Master: - S/W should program this bit only after transmission of an ICLC frame changing the speed mode of the slaves Rx interface. For LFAST Slave: - When SCR[DRMD] = 1, the H/W programs this bit on reception of ICLC frame for changing speed mode of Slaves Tx interface. - The S/W can program this bit when SCR[DRMD] = 0. - This bit is cleared on MCR[DRFEN] negation. 0 Data rate of Tx block is low speed. 1 Data rate of Tx block is high speed.

45.7.3 LFAST Correlator Control Register (LFAST_COCR)

The COCR is used to select the sampler data path and the number of bits of correlation to be used.

Address: 0h base + 8h offset = 8h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	SMPSEL								0						0	
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	
R	0												CORRTH				PHSEL
W																	
Reset	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	0	

LFAST_COCR field descriptions

Field	Description
0–7 SMPSEL	Sampler Data Path Selector (overrides the correlator selection). Defines the sampler data path to be activated at all the times. All the bits should be 0 (00h) for Sampler Data Path to be selected by the correlator. In Low Speed mode only Sampler Data Paths 0-3 are valid. This field can only be written when MCR[RXEN] = 0.

Table continues on the next page...

LFAST_COCR field descriptions (continued)

Field	Description
	00h Sampler Data Path selected by correlator 01h Sampler Data Path 0 selected 02h Sampler Data Path 1 selected 04h Sampler Data Path 2 selected 08h Sampler Data Path 3 selected 10h Sampler Data Path 4 selected 20h Sampler Data Path 5 selected 40h Sampler Data Path 6 selected others Sampler Data Path 7 selected
8–13 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
14–15 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
16–27 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
28–30 CORRTH	Correlator threshold level. Defines the correlation threshold level. This field can only be written when MCR[RXEN] = 0. 000 9 Bits of correlation 001 10 Bits of correlation ... 110 15 Bits of correlation 111 16 Bits of correlation
31 PHSSEL	Polyphase 8 or 4 phase selection. Defines the number of phases for the polyphase generator used. This bit is ignored in low speed mode since only 4 Phases are used in low speed mode. In High Speed mode phase 0, 2, 4 and 6 are used when 4 phase mode is selected. This field can only be written when MCR[RXEN] = 0 0 8 phases 1 4 phases

45.7.4 LFAST Test Mode Control Register (LFAST_TMCR)

The TMCR enables and configures the LFAST clock test and loopback modes.

Address: 0h base + Ch offset = Ch

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0						CLKTST	LPON	0						LPMOD	
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	LPFRMTH															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

LFAST_TMCR field descriptions

Field	Description
0–5 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
6 CLKTST	Clock Test mode. S/W can define when the clock test mode is enabled. This bit can also be set and cleared by LFAST slave H/W on reception of an ICLC command. This field can only be written when MCR[DRFEN] = 1. 1 Clock Test mode enabled 0 Clock Test mode disabled
7 LPON	Loopback mode Logic Enable. S/W can define when the loopback logic is enabled. This bit can also be written by LFAST slave H/W on reception of an ICLC command. This field can only be written when MCR[DRFEN] = 1. 1 Loopback mode is enabled 0 Loopback mode is disabled
8–12 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
13–15 LPMOD	Loopback mode. Defines the type of loopback mode enabled. This field can only be written when TMCR[LPON] = 0. 000 Rx loopback 001 Rx LVDS loopback 010 Tx loopback without automatic frame generation 011 Tx loopback with automatic frame generation 100 Tx LVDS loopback (external) with automatic frame generation 101 Reserved

Table continues on the next page...

LFAST_TMCR field descriptions (continued)

Field	Description
	110 Reserved 111 Reserved
16–31 LPFRMTH	Loopback check mode valid pass frames threshold value. Defines the number of frames to verify before setting GCR[LPFPDV] when running in Automatic Loopback Frame mode. The loopback frame is considered pass when the payload is CBh, header is 13h and sync is valid. This mode is valid only when TMCR[LPMOD] = 011b or TMCR[LPMOD] = 100b. This field can only be written when TMCR[LPON] = 0. 0000h Reserved.(Not to be used) 0001h Check 1 frame have correct sync, header and payload. ... FFFEh Check 65534 frames have correct sync, header and payload. FFFFh Check 65535 frames have correct sync, header and payload.

45.7.5 LFAST Auto Loopback Control Register (LFAST_ALCR)

Address: 0h base + 10h offset = 10h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0														LPCNTEN	
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	LPFMCNT															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

LFAST_ALCR field descriptions

Field	Description
0–14 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
15 LPCNTEN	Auto Loopback Frame Transmission Count Enable. Enables fixed number of auto pre-defined loopback frame transmission. This field can only be written when TMCR[LPON] = 0. 0 Infinite pre-defined loopback frame transmission enabled 1 Fixed count of pre-defined loopback frame transmission enabled
16–31 LPFMCNT	Auto Loopback Frame Transmission Count. Defines the number of pre-defined auto loopback frames to be sent. The pre-defined loopback frame has payload CBh, header 13h and valid sync. This mode is valid if TMCR[LPMOD] = 011b or TMCR[LPMOD] = 100b. This field can only be written when TMCR[LPON] = 0. 0000h Reserved.(Not to be used) 0001h Send 1 frame with correct sync, header and payload

Table continues on the next page...

LFAST_ALCR field descriptions (continued)

Field	Description
...	FFFEh Send 65534 frames with correct sync, header and payload
FFFFh	Send 65535 frames with correct sync, header and payload

45.7.6 LFAST Rate Change Delay Control Register (LFAST_RCDCR)

Address: 0h base + 14h offset = 14h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0												DRCNT				0															
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	

LFAST_RCDCR field descriptions

Field	Description
0–11 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
12–15 DRCNT	Data Rate Controller Counter Value. Defines the number of cycles of Phase 0 clock needed by the Tx interface Data rate change controller to switch from one speed mode to another. The arbitrator ignores all requests during this period. This field can only be written when MCR[DRFEN] = 1. Number of cycles = DRCNT value.
16–31 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

45.7.7 LFAST Wakeup Delay Control Register (LFAST_SLCR)

Address: 0h base + 18h offset = 18h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31			
R	HSCNT								0				LSCNT				HWKCNT								0				LWKCNT						
W																																			
Reset	0	0	0	1	0	0	1	0	0	0	0	0	0	0	0	1	0	1	0	1	1	1	1	1	0	0	0	0	0	0	1	0			

LFAST_SLCR field descriptions

Field	Description
0–7 HSCNT	High Speed Sleep mode Exit Time. Defines 1/4 of the number of the high speed clock cycle wait after the negation of the LVDS LD sleep signal, and before the start of new frame. This wait is the summation of settling time of the Line Driver (LD) after negation of its sleep signal, and the wakeup time of the LR of the peer LFAST. This field can only be written when MCR[DRFEN] = 0. 00h 0 cycle 01h 1 cycle ...

Table continues on the next page...

LFAST_SLCR field descriptions (continued)

Field	Description
	12h 18 cycles (200 ns + 8 cycles of High speed clock) ... FEh 254 cycles FFh 255 cycles
8–11 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
12–15 LSCNT	Low Speed Sleep mode Exit Time. Defines 1/4 of the number of Low speed clock cycle wait after the negation of LVDS LD sleep signal, and before the start of new frame. This wait is the summation of settling time of LD after negation of LVDS LD sleep signal, and the wakeup time of the LR of the peer LFAST. This field can only be written when MCR[DRFEN] = 0. 0h 0 cycle 1h 1 cycle (200ns + 1 cycle of Low speed clock) ... Eh 14 cycles Fh 15 cycles
16–23 HWKCNT	Wake Up time for the LD. Defines the 1/4 of the number of High speed clock cycles used during the wakeup of the LD from shutdown mode. This is time required by the LD to move from moving from Shutdown to Normal State in High speed mode. This field can only be written when MCR[DRFEN] = 0. 00h 0 cycles 01h 1 cycle ... 5Fh 95 cycles (1.18 μ s) ... FFh 255 cycles Maximum
24–27 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
28–31 LWKCNT	Wake Up time for the LD. Defines the 1/4 of the number Low speed clock used during the wakeup of the LD from shutdown mode. This is time required by the LD to move from Shutdown to Normal State in Low speed mode. This field can only be written when MCR[DRFEN] = 0. 0h 0 cycle 1h 1 cycle 2h 2 cycles (1.18 μ s) ... Eh 14 cycles Fh 15 cycles

45.7.8 LFAST ICLC Control Register (LFAST_ICR)

Address: 0h base + 1Ch offset = 1Ch

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0														ICLCSEQ	SNDICLC
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	Reserved								ICLCPLD							
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

LFAST_ICR field descriptions

Field	Description
0–13 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
14 ICLCSEQ	ICLC enabled. This bit should be set whenever the S/W is performing a series of ICLC frame transfers. This bit ensures only ICLC frame transmission request is serviced, other pending frames request are ignored. 0 Single ICLC frame transfer. 1 S/W is performing ICLC frame transfers. Only the ICLC frames will be scheduled during this period. All the other frames will be scheduled only after ICR[ICLCSEQ] = 0.
15 SNDICLC	ICLC frame request. This bit is set to initiate the transfer of ICLC frame by LFAST master. This bit should be set (ICR[SNDICLC] = 1) after writing the required ICLC payload to be transmitted in the ICLCPLD field of the register. This bit is self clearing, which will be cleared when ICLC frame transfer is complete. Set by user software and cleared by system hardware. 0 No Valid ICLC frame for transfer. 1 Valid ICLC frame for transfer.
16–23 Reserved	This field is reserved.
24–31 ICLCPLD	ICLC Payload. This field is used to program the payload of the ICLC frame to be transmitted. New ICLC payload should be set when ICR[SNDICLC] = 0. This field can only be written when ICR[SNDICLC] = 0 (see Table 45-53 for supported ICLC payloads).

45.7.9 LFAST Ping Control Register (LFAST_PICR)

Address: 0h base + 20h offset = 20h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	
R	0														PNGREQ		
W																	
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	
R	PNGAUTO	0							PNGPYLD								
W																	
Reset	1	0	0	0	0	0	0	0	1	1	0	0	1	0	1	0	

LFAST_PICR field descriptions

Field	Description
0–14 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
15 PNGREQ	Ping Response Frame Request. This bit is set to initiate the transfer of Ping response frame. Cleared after transmission of Ping response frame. Set by user software and cleared by system hardware. This field can only be written when MCR[DRFEN] = 1. 1 Ping response frame transmission request is queued 0 No pending Ping response frame transmission request
16 PNGAUTO	Ping Response Enable. Defines when ping response should be automatically sent on reception of Ping ICLC frame from LFAST master. This field can only be written when MCR[DRFEN] = 0. 0 Ping response should not be automatically sent. 1 Ping response should be automatically sent.
17–23 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
24–31 PNGPYLD	LFAST Slave: Defines the LFAST slaves ping reply frame payload content. This field can only be written when MCR[DRFEN] = 0. LFAST Master: Defines the expected payload of ping response frame. Used for comparison with ping frame received.

45.7.10 LFAST Rx FIFO CTS Control Register (LFAST_RFCR)

Address: 0h base + 2Ch offset = 2Ch

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0									RCTSMX						0									RCTSMN							
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	1

LFAST_RFCR field descriptions

Field	Description
0–9 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
10–15 RCTSMX	Rx FIFO Maximum Threshold. Defines the condition for CTS bit of frames to be negated. This field can only be written when LFAST_MCR[DRFEN] = 0.
16–25 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
26–31 RCTSMN	Rx FIFO Minimum Threshold. Defines the condition for CTS bit of frames to be set. This field can only be written when LFAST_MCR[DRFEN] = 0.

45.7.11 LFAST Tx Interrupt Enable Register (LFAST_TIER)

Address: 0h base + 30h offset = 30h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0													TXIIE	TXOVIE	
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	
R	0												TXPNGIE	0	TXUNSIE	TXICLCIE	TXDITIE
W																	
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	

LFAST_TIER field descriptions

Field	Description
0–13 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
14 TXIIE	Tx Data Interface Not Enabled - (Mask) Enables or disables the interrupt. Tx Data Interface not enabled and a frame is ready to be transmitted

Table continues on the next page...

LFAST_TIER field descriptions (continued)

Field	Description
	0 Interrupt is disabled 1 Interrupt is enabled
15 TXOVIE	Transmit Data FIFO Overflow Interrupt Enable. 0 Interrupt is disabled 1 Interrupt is enabled
16–26 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
27 TXPNGIE	Ping Response Frame Transmitted Interrupt Enable. 0 Interrupt is disabled 1 Interrupt is enabled
28 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
29 TXUNSIE	Unsolicited Frame transmitted Interrupt Enable 0 Interrupt is disabled 1 Interrupt is enabled
30 TXICLCIE	ICLC Frame transmitted Interrupt Enable 0 Interrupt is disabled 1 Interrupt is enabled
31 TXDTIE	Data Frame transmitted Interrupt Enable 0 Interrupt is disabled 1 Interrupt is enabled

45.7.12 LFAST Rx Interrupt Enable Register (LFAST_RIER)

Address: 0h base + 34h offset = 34h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0								RXUOIE	RXMNIE	RXMXIE	RXUFIE	RXOFIE	RXSZIE	RXICIE	RXLCEIE
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0												RXCTSIE	RXDIE	RXUNSIE	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

LFAST_RIER field descriptions

Field	Description
0–7 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
8 RXUOIE	Unsolicited frame register overflow. Indicates existing unsolicited frame hasn't been read and a new unsolicited frame has arrived. 0 Interrupt is disabled 1 Interrupt is enabled
9 RXMNIE	Rx Data FIFO Min Threshold reached 0 Interrupt is disabled 1 Interrupt is enabled
10 RXMXIE	Rx Data FIFO Max Threshold reached 0 Interrupt is disabled 1 Interrupt is enabled
11 RXUFIE	Rx Data FIFO Underflow 0 Interrupt is disabled 1 Interrupt is enabled
12 RXOFIE	Rx Data FIFO Overflow 0 Interrupt is disabled 1 Interrupt is enabled
13 RXSZIE	Frame with unsupported frame size received. Valid frame sizes are defined in Table 45-57 0 Interrupt is disabled 1 Interrupt is enabled
14 RXICIE	Invalid ICLC code Received 0 Interrupt is disabled 1 Interrupt is enabled
15 RXLCEIE	Invalid Logical Channel Type 0 Interrupt is disabled 1 Interrupt is enabled
16–27 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
28 RXCTSIE	Frame with CTS bit Low Received 0 Interrupt is disabled 1 Interrupt is enabled
29 RXDIE	Data frame received 0 Interrupt is disabled 1 Interrupt is enabled
30 RXUNSIE	Unsolicited Frame received 0 Interrupt is disabled 1 Interrupt is enabled

Table continues on the next page...

LFAST_RIER field descriptions (continued)

Field	Description
31 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

45.7.13 LFAST Rx ICLC Interrupt Enable Register (LFAST_RIER)

Address: 0h base + 38h offset = 38h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0		ICPFIE	ICPSIE	ICPRIIE	ICTOIE	ICLPIE	ICCTIE	ICTDIE	ICTEIE	ICRFIE	ICRSIE	ICTFIE	ICTSIE	ICPOFIE	ICPONIE
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

LFAST_RIER field descriptions

Field	Description
0–17 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
18 ICPFIE	Ping Frame Response failed 0 Interrupt is disabled 1 Interrupt is enabled
19 ICPSIE	Ping Frame Response successful 0 Interrupt is disabled 1 Interrupt is enabled
20 ICPRIIE	ICLC frame for Ping Frame Request received 0 Interrupt is disabled 1 Interrupt is enabled
21 ICTOIE	ICLC frame for Test mode off received 0 Interrupt is disabled 1 Interrupt is enabled
22 ICLPIE	ICLC frame for Loopback On received 0 Interrupt is disabled 1 Interrupt is enabled

Table continues on the next page...

LFAST_RIIER field descriptions (continued)

Field	Description
23 ICCTIE	ICLC frame for Clk Test mode on received 0 Interrupt is disabled 1 Interrupt is enabled
24 ICTDIE	ICLC frame for LFAST Slaves Tx Interface Disable received 0 Interrupt is disabled 1 Interrupt is enabled
25 ICTEIE	ICLC frame for LFAST Slaves Tx Interface Enable received 0 Interrupt is disabled 1 Interrupt is enabled
26 ICRFIE	ICLC frame for LFAST Slaves Rx Interface fast mode switch received 0 Interrupt is disabled 1 Interrupt is enabled
27 ICRSIE	ICLC frame for LFAST Slaves Rx Interface slow mode switch received 0 Interrupt is disabled 1 Interrupt is enabled
28 ICTFIE	ICLC frame for LFAST Slaves Tx Interface fast mode switch received 0 Interrupt is disabled 1 Interrupt is enabled
29 ICTSIE	ICLC frame for LFAST Slaves Tx Interface slow mode switch received 0 Interrupt is disabled 1 Interrupt is enabled
30 ICPOFIE	ICLC frame for PLL OFF received 0 Interrupt is disabled 1 Interrupt is enabled
31 ICPONIE	ICLC frame for PLL ON received 0 Interrupt is disabled 1 Interrupt is enabled

45.7.14 LFAST PLL Control Register (LFAST_PLLCR)

Address: 0h base + 3Ch offset = 3Ch

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	IPTMOD			0											SWPOFF	SWPON
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31				
R	REFINV	Reserved		0	PLCKCW			FDIVEN	FBDIV										PREDIV	
W																				
Reset	0	0	0	0	0	0	0	0	0	1	0	1	1	1	0	0				

LFAST_PLLCR field descriptions

Field	Description
0–2 IPTMOD	Test mode programmability 000 Functional mode 001 Closed Loop 1 010 Force Vctrl 011 Charge Pump Up 100 Charge Pump Up Internal Test 101 Charge Pump Idle 110 Charge Pump Down 111 Closed Loop 2
3–13 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
14 SWPOFF	SW signal to turn OFF the PLL. Set by user software and cleared by system hardware. 0 No effect 1 PLL will be turned OFF.
15 SWPON	SW signal to turn ON the PLL. Set by user software and cleared by system hardware. 0 No effect 1 PLL will be turned ON
16 REFINV	Inverts reference clock edge to PFD. If System PLL PFD using same reference clock, enabling this feature will minimize noise injection crosstalk via the substrate. 0 Do not inverted 1 Invert

Table continues on the next page...

LFAST_PLLCR field descriptions (continued)

Field	Description
17–18 Reserved	This field is reserved.
19–20 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
21–22 PLCKCW	PLL Lock Ready Count Width. Defines the number of cycles PLL waits for before asserting lock_ready flag High 00 1040 cycles 01 520 cycles 10 320 cycles 11 200 cycles
23 FDIVEN	Enable fraction division mode in feedback divider. Enables the division of vco clock output by a factor of (FDIV + 0.5). 0 Fraction division mode not enabled 1 Fraction division mode enabled
24–29 FBDIV	Feedback Division factor for VCO output clock. This field can only be written when LFAST_MCR[DRFEN] = 0. 00h No clock output 01h – 0Ah Reserved 0Bh Divide by 12 (11.5, if FDIVEN = 1) ... 1Fh Divide by 32 (31.5, if FDIVEN = 1) 20h – 3Fh Reserved
30–31 PREDIV	Division factor for PLL Reference Clock input. This field can only be written when LFAST_MCR[DRFEN] = 0. 00 Direct clock passed 01 Divide by 2 10 Divide by 3 11 Divide by 4

45.7.15 LFAST LVDS Control Register (LFAST_LCR)

Address: 0h base + 40h offset = 40h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0								SWWKLD	SWSLPLD	SWWKLR	SWSLPLR	SWOFFLD	SWONLD	SWOFFLR	SWONLR
W																
Reset	0	0	0	0	0	0	0	0	0*	0*	0*	0*	0*	0*	0*	0*

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	LVRXOFF	LVTXOE	TXCMUX	LVRFEN	LVLPEN	0					LVRXOP_TR	Reserved	LVRXOP_BR	LVTXOP	LVCKSS	LVCKP
W																
Reset	0	1	0	1	0	0	0	0	0	0	1	0	0	1	0	0

* Notes:

- SWONLR field: Set by user software and cleared by system hardware.
- SWOFFLR field: Set by user software and cleared by system hardware.
- SWONLD field: Set by user software and cleared by system hardware.
- SWOFFLD field: Set by user software and cleared by system hardware.
- SWSLPLR field: Set by user software and cleared by system hardware.
- SWWKLR field: Set by user software and cleared by system hardware.
- SWSLPLD field: Set by user software and cleared by system hardware.
- SWWKLD field: Set by user software and cleared by system hardware.

LFAST_LCR field descriptions

Field	Description
0–7 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
8 SWWKLD	SW signal to take LVDS LD out of Sleep mode. This field can only be written when LFAST_MCR[DRFEN] = 1. NOTE: Writes are not reflected in the read of this field. 0 No effect 1 LVDS LD will be taken out of sleep (provided no other source is trying to put it in sleep)
9 SWSLPLD	SW signal to put LVDS LD into Sleep mode. This field can only be written when LFAST_MCR[DRFEN] = 1. NOTE: Writes are not reflected in the read of this field. 0 No effect 1 LVDS LD will be put in sleep

Table continues on the next page...

LFAST_LCR field descriptions (continued)

Field	Description
10 SWWKLR	SW signal to take LVDS LR out of Sleep mode. This field can only be written when LFAST_MCR[DRFEN] = 1. NOTE: Writes are not reflected in the read of this field. 0 No effect 1 LVDS LR will be taken out of sleep (provided no other source is trying to put it in sleep)
11 SWSLPLR	SW signal to put LVDS LR into Sleep mode. This field can only be written when LFAST_MCR[DRFEN] = 1. NOTE: Writes are not reflected in the read of this field. 0 No effect 1 LVDS LR will be put in sleep (provided no other source is trying to wake it up)
12 SWOFFLD	SW signal to turn OFF the LVDS LD. This field can only be written when LFAST_MCR[DRFEN] = 1. NOTE: Writes are not reflected in the read of this field. 0 No effect 1 LVDS LD will be turned OFF(provided no other source is trying to turn the LD ON)
13 SWONLD	SW signal to turn ON the LVDS LD. This field can only be written when LFAST_MCR[DRFEN] = 1. NOTE: Writes are not reflected in the read of this field. 0 No effect 1 LVDS LD will be turned ON
14 SWOFFLR	SW signal to turn OFF the LVDS LR. This field can only be written when LFAST_MCR[DRFEN] = 1. NOTE: Writes are not reflected in the read of this field. 0 No effect 1 LVDS LR will be turned OFF (provided no other source is trying to turn the LR ON)
15 SWONLR	SW signal to turn ON the LVDS LR. This field can only be written when LFAST_MCR[DRFEN] = 1. NOTE: Writes are not reflected in the read of this field. 0 No effect 1 LVDS LR will be turned ON
16 LVRXOFF	Indicates the value driven onto LVDS LR output when in shutdown mode
17 LVTXOE	LVDS LD output buffer enable. 0 LVDS LD output buffer enable is disabled. 1 LVDS LD output buffer enabled
18 TXCMUX	Tx and Clock Mux. The bit can be used to bring out PLL Phase 0 clock on Tx LVDS pad. 0 No effect 1 PLL Phase 0 clock will be brought out to Tx LVDS pad
19 LVRFEN	LVDS pad reference enable 0 LVDS reference pad disabled 1 LVDS reference pad enabled

Table continues on the next page...

LFAST_LCR field descriptions (continued)

Field	Description
20 LVLPEN	Tx LVDS internal loopback enable The bit is reflected by output signal <code>ipp_digrf_lvds_lpbk_en</code>. The internal loopback is not intended for board level functionality, so this feature should not be implemented. 0 Tx LVDS normal mode enabled 1 Tx LVDS internal loopback mode enabled
21–25 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
26 LVRXOP_TR	Used to enable or disable the on-chip receiver termination resistor in LFAST mode (applies to LVDS pad use for LFAST only): 0 Disable on-chip LFAST receiver termination 1 Enable on-chip LFAST receiver termination
27 Reserved	This field is reserved.
28 LVRXOP_BR	Used to set the bias current for the receiver in LFAST mode. It is recommended to always write 1 to this bit when using the LFAST interface. Writing 0 will allow for a small power savings during lower baud rates (applies to LVDS pad use for LFAST only): 0 Use for LFAST receiver baud rates less than the maximum baud rate. 1 Required for LFAST receiver maximum baud rate.
29 LVTXOP	Control signal for LFAST and Micro-Second Bus selection. 0 Micro-Second Bus selection 1 LFAST Bus selection
30 LVCKSS	LVDS Clock Sync Select This bit is used to adjust the LVDS data sampling to the duty cycle of the clock. It is recommended that a value of zero is used in all cases. A value of one is reserved for specific devices/revisions with different clock timing. 0 normal clock used to sample the LVDS data 1 adjusted clock used to sample the LVDS data
31 LVCKP	LVDS clock select. This bit is used for selecting use of direct pll clock or inverted pll clock in LVDS. 0 Direct pll clock to be used inside the LVDS 1 Inverted pll clock to be used inside the LVDS

45.7.16 LFAST Unsolicited Tx Control Register (LFAST_UNSTCR)

Address: 0h base + 44h offset = 44h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0														USNDRQ	
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0								UNSHDR							
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

LFAST_UNSTCR field descriptions

Field	Description
0–14 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
15 USNDRQ	Tx Unsolicited send request. Set by user software and cleared by system hardware. This field can only be written when MCR[DRFEN] = 1. 0 No valid Unsolicited frame exists 1 Valid Unsolicited frame exists for transmission
16–24 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
25–31 UNSHDR	Tx Unsolicited message header. This field can only be written when LFAST_UNSTCR[USNDRQ] = 0

45.7.17 LFAST Unsolicited Tx Data Registers (LFAST_UNSTDR_n)

Address: 0h base + 48h offset + (4d × i), where i=0d to 8d

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	UNTXD																															
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

LFAST_UNSTDR_n field descriptions

Field	Description
0–31 UNTXD	Unsolicited Transmit Data 8-0. This represents 9 registers for Unsolicited transmit data. The first bit to transmitted as part of the payload will be from UNTXD8[31], second bit from UNTXD8[30], and so on. So the last bit transmitted will be from UNTXD0[0] in case of 288 bit payload.

45.7.18 LFAST Global Status Register (LFAST_GSR)

Address: 0h base + 80h offset = 80h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	DUALMD	0												LRMD	LDSM	DRSM
W																
Reset	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0											LPTXDN	LPFPDV	LPCPDV	LPCHDV	LPCSDV
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

LFAST_GSR field descriptions

Field	Description
0 DUALMD	Indicates the LFAST module is in Dual mode

Table continues on the next page...

LFAST_GSR field descriptions (continued)

Field	Description
	0 LFAST Module in Slave only mode 1 LFAST Module in Dual mode
1–12 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
13 LRMD	Indicates if the Rx Controller is idle/active and that the Rx clocks are enabled. In functional mode this will always be active. 0 Rx Controller is in Idle state 1 Rx Controller is active
14 LDSM	Transmit Interface Data Rate Status. Indicates the current speed rate of the Tx controller 0 Data rate of LOW speed mode 1 Data rate of HIGH speed mode
15 DRSM	Receive Interface Data Rate Status. Indicates the current speed rate of the Rx controller 0 Data rate of LOW speed mode 1 Data rate of HIGH speed mode
16–26 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
27 LPTXDN	Auto loopback frame transmission count reached. The Count of frame is defined by LFAST_ALCR[LPFMCNT]. 0 Auto loopback frame transmission count not reached. 1 Auto loopback frame transmission count reached.
28 LPFPDV	Loopback frame pass threshold reached. 0 Pass frame threshold not reached. 1 Pass frame threshold achieved
29 LPCPDV	Valid payload received during loopback check mode. Indicates whether the Loop back frame received payload is CBh. 0 Payload received is not CBh 1 Payload received is CBh
30 LPCHDV	Valid header received during loopback check mode. Indicates whether the Loop back frame received header is 13h. 0 Header received is not 13h 1 Header received is 13h
31 LPCSDV	Valid synchronization received. 0 Valid Synchronization pattern not detected 1 Valid Synchronization pattern detected

45.7.19 LFAST Ping Status Register (LFAST_PISR)

Address: 0h base + 84h offset = 84h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0																							RXPNGD								
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	

LFAST_PISR field descriptions

Field	Description
0–23 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
24–31 RXPNGD	Ping Data Register. In LFAST Master mode the Ping response ICLC frame received is stored into this register.

45.7.20 LFAST Data Frame Status Register (LFAST_DFSR)

Address: 0h base + 94h offset = 94h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	RXDCNT						0					RXFCNT			0	TXDCNT						0					TXFCNT				
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

LFAST_DFSR field descriptions

Field	Description
0–1 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
2–7 RXDCNT	Unread Rx Frame Data Count. Indicates the number of unread data stored in the Rx Data FIFO.
8–12 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
13–15 RXFCNT	Unread Rx Frame Count. Indicates the number of unread data frames stored in the Rx Data FIFO.
16–17 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
18–23 TXDCNT	Unread Tx Frame Data Count. Indicates the number of unread data stored in the Tx Data FIFO.
24–28 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
29–31 TXFCNT	Unread Tx Frame Count.Count of pending Data Frames programmed by System Side Module.

45.7.21 LFAST Tx Interrupt Status Register (LFAST_TISR)

Address: 0h base + 98h offset = 98h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0														TXIEF	TXOVF
W															w1c	w1c
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0										TXPNGF		0	TXUNSF	TXICLCF	TXDTF
W											w1c			w1c	w1c	w1c
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

LFAST_TISR field descriptions

Field	Description
0–13 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
14 TXIEF	TxData Interface not enabled. Tx Data Interface not enabled and a frame is ready to be transmitted 0 Interrupt event has not occurred 1 Interrupt event has occurred
15 TXOVF	Transmit Data FIFO Overflow Interrupt 0 Interrupt event has not occurred 1 Interrupt event has occurred
16–26 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
27 TXPNGF	Ping response frame transmitted interrupt 0 Interrupt event has not occurred 1 Interrupt event has occurred
28 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

Table continues on the next page...

LFAST_TISR field descriptions (continued)

Field	Description
29 TXUNSF	Unsolicited Frame transmitted Interrupt 0 Interrupt event has not occurred 1 Interrupt event has occurred
30 TXICLCF	ICLC Frame transmitted Interrupt 0 Interrupt event has not occurred 1 Interrupt event has occurred
31 TXDTF	Data Frame transmitted Interrupt 0 Interrupt event has not occurred 1 Interrupt event has occurred

45.7.22 LFAST Rx Interrupt Status Register (LFAST_RISR)

Address: 0h base + 9Ch offset = 9Ch

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0								RXUOF	RXMNF	RXMXF	RXUFF	RXOFF	RXSZF	RXICF	RXLCEF
W									w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0												RXCTSF	RXDF	RXUNSF	0
W													w1c	w1c	w1c	
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

LFAST_RISR field descriptions

Field	Description
0–7 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

Table continues on the next page...

LFAST_RISR field descriptions (continued)

Field	Description
8 RXUOF	Unsolicited frame register overflow. Indicates existing unsolicited frame hasn't been read and a new unsolicited frame has arrived. 0 Interrupt event has not occurred 1 Interrupt event has occurred
9 RXMNF	Rx Data FIFO Min Threshold reached. 0 Interrupt event has not occurred 1 Interrupt event has occurred
10 RXMXF	Rx Data FIFO Max Threshold reached. 0 Interrupt event has not occurred 1 Interrupt event has occurred
11 RXUFF	Rx Data FIFO Underflow. 0 Interrupt event has not occurred 1 Interrupt event has occurred
12 RXOFF	Rx Data FIFO Overflow. 0 Interrupt event has not occurred 1 Interrupt event has occurred
13 RXSZF	Frame with unsupported frame size received. See "Frames Supported by LFAST interfaces" table for details. 0 Interrupt event has not occurred 1 Interrupt event has occurred - On reception of frame with payload size for a frame other than mentioned in the "Frames Supported by LFAST interfaces" table.
14 RXICF	Invalid ICLC code Received. 0 Interrupt event has not occurred 1 Interrupt event has occurred
15 RXLCEF	Invalid Logical Channel Type. 0 Interrupt event has not occurred 1 Interrupt event has occurred - On reception of frame other than mentioned in the "Frames Supported by LFAST interfaces" table.
16–27 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
28 RXCTSF	Frame with CTS bit Low Received. 0 Interrupt event has not occurred 1 Interrupt event has occurred
29 RXDF	Data frame received. 0 Interrupt event has not occurred 1 Interrupt event has occurred
30 RXUNSF	Unsolicited Frame received. 0 Interrupt event has not occurred 1 Interrupt event has occurred

Table continues on the next page...

LFAST_RISR field descriptions (continued)

Field	Description
31 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

45.7.23 LFAST Rx ICLC Interrupt Status Register (LFAST_RISR)

Address: 0h base + A0h offset = A0h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0		ICPFF	ICPSF	ICPRF	ICTOF	ICLPF	ICCTF	ICTDF	ICTEF	ICRFF	ICRSF	ICTFF	ICTSF	ICPOFF	ICPONF
W			w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

LFAST_RISR field descriptions

Field	Description
0–17 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
18 ICPFF	Ping Frame Response failed 0 Interrupt event has not occurred 1 Interrupt event has occurred
19 ICPSF	Ping Frame Response successful 0 Interrupt event has not occurred 1 Interrupt event has occurred
20 ICPRF	ICLC Ping Frame Request received

Table continues on the next page...

LFAST_RIISR field descriptions (continued)

Field	Description
	0 Interrupt event has not occurred 1 Interrupt event has occurred
21 ICTOF	ICLC frame for Test mode off received 0 Interrupt event has not occurred 1 Interrupt event has occurred
22 ICLPF	ICLC frame for Loopback On received 0 Interrupt event has not occurred 1 Interrupt event has occurred
23 ICCTF	ICLC frame for Clk Test mode received 0 Interrupt event has not occurred 1 Interrupt event has occurred
24 ICTDF	ICLC frame for LFAST Slaves Tx Interface Disable received 0 Interrupt event has not occurred 1 Interrupt event has occurred
25 ICTEF	ICLC frame for LFAST Slaves Tx Interface Enable received 0 Interrupt event has not occurred 1 Interrupt event has occurred
26 ICRFF	ICLC frame for LFAST Slaves Rx Interface fast mode switch received 0 Interrupt event has not occurred 1 Interrupt event has occurred
27 ICRSF	ICLC frame for LFAST Slaves Rx Interface slow mode switch received 0 Interrupt event has not occurred 1 Interrupt event has occurred
28 ICTFF	ICLC frame for LFAST Slaves Tx Interface fast mode switch received 0 Interrupt event has not occurred 1 Interrupt event has occurred
29 ICTSF	ICLC frame for LFAST Slaves Tx Interface slow mode switch received 0 Interrupt event has not occurred 1 Interrupt event has occurred
30 ICPOFF	ICLC frame for PLL OFF received 0 Interrupt event has not occurred 1 Interrupt event has occurred
31 ICPONF	ICLC frame for PLL ON received 0 Interrupt event has not occurred 1 Interrupt event has occurred

45.7.24 LFAST PLL and LVDS Status Register (LFAST_PLLLSR)

Address: 0h base + A4h offset = A4h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	
R	Reserved													PLLDIS	PLDCR		
W																	
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	
Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	
R	0													LRS LPS	LDS LPS	LDPDS	LRPDS
W																	
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	

LFAST_PLLLSR field descriptions

Field	Description
0–13 Reserved	This field is reserved.
14 PLLDIS	PLL disable Status. When asserted, PLL is put in the power down state. 0 PLL disable signal is negated. 1 PLL disable signal is asserted.
15 PLDCR	PLL Lock Delay Counter Ready. When asserted this bit indicates that the PLL is locked after N number of reference/PLLCR[PREDIV] cycles 0 PLL Lock delay counter is not decremented to 0 1 PLL Lock delay counter is decremented to 0

Table continues on the next page...

LFAST_PLLLSR field descriptions (continued)

Field	Description
16–27 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
28 LRSLPS	This bit indicates the real time status of LR sleep signal 0 LR power sleep signal is negated. 1 LR power sleep signal is asserted.
29 LDLPS	This bit indicates the real time status of LD sleep signal 0 LD sleep signal is negated. 1 LD sleep signal is asserted.
30 LDPDS	This bit indicates the real time status of LD power down signal When asserted, LD is put in the power down state. 0 LD power down signal is negated. 1 LD power down signal is asserted.
31 LRPDS	This bit indicates the real time status of LR power down signal When asserted, LR is put in the power down state. 0 LR power down signal is negated. 1 LR power down signal is asserted.

45.7.25 LFAST Unsolicited Rx Status Register (LFAST_UNSRSR)

Address: 0h base + A8h offset = A8h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0							URXDV	0							URPCNT
W								0								
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

LFAST_UNSRSR field descriptions

Field	Description
0–22 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
23 URXDV	Unsolicited data valid. Indicates a valid frame exists in the Unsolicited Data registers. NOTE: Writing 0 will clear this bit
24–28 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
29–31 URPCNT	Rx Unsolicited payload. Indicates the number of bytes of valid Rx unsolicited Data payload present in the LFAST_UNSRDR[8:0].

45.7.26 LFAST Unsolicited Rx Data Register (LFAST_UNSRDR_n)

Address: 0h base + ACh offset + (4d × i), where i=0d to 8d

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	UNRXD																															
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

LFAST_UNSRDR_n field descriptions

Field	Description
0–31 UNRXD	Unsolicited Receive Data. This represents 9 registers for Unsolicited received data. It is read only register. The first bit received as part of the payload will be stored at UNRXD8[31], second bit at UNRXD8[30], and so on. So the last bit will be stored at UNRXD0[0] in case of 288 bit payload.

45.8 Functional description**45.8.1 Startup procedure**

The LFAST requires a sequence of operations before it can be used for communication. The procedure to enable LFAST for proper operation is listed below.

45.8.1.1 LFAST master interface startup procedure

1. After reset the SLCR and RCDRCR are programmed according to the LVDS parameters in the device data sheet.

2. The PLLCR is programmed with configuration parameters for the PLL.
3. The LCR is programmed with configuration parameters of the LVDS.
4. Write MCR[MSEN] = 1. Then select LFAST modes by configuring MCR[CTSEN] and MCR[DATAEN].
5. Write MCR[DRFEN] = 1 to enable the LFAST.
6. Write MCR[RXEN] = 1 and MCR[TXEN] = 1 to negate the LD powerdown, LR disable and LR powerdown signals.
7. Write ICR[ICLCPLD] = 31h to enable the Slaves Tx Interface.
8. Write ICR[SNDICLC] = 1 and ICR[ICLCSEQ] = 1.
9. Write MCR[TXARBD] = 0.
10. The ICLC transmission is confirmed by verifying one of the following:
 - ICR[SNDICLC] = 0.
 - TISR[TXICLCF] = 1.
11. Write ICR[ICLCPLD] = 00h to check the LFAST slaves status.
12. Write ICR[SNDICLC] = 1.
13. The LFAST slave status is confirmed by occurrence of one of the following:
 - LFAST slave is enabled if RIISR[ICPSF] = 1. Proceed to step 14.
 - LFAST slave is disabled if RIISR[ICPFF] = 1. The LFAST master must wait and restart from Step 7.

Speed mode change:

14. Write PLLCR[SWPON] = 1 to enable the LFAST masters PLL.
15. Write ICLC start PLL frame, ICR[ICLCPLD] = 02h.
16. Write ICR[SNDICLC] = 1.
17. The ICLC transmission is confirmed by occurrence of one of the following:
 - ICR[SNDICLC] = 0.
 - TISR[TXICLCF] = 1.
18. To change LFAST masters Tx interface speed both of the following operations are performed:

Note

(The slaves Rx interface speed should be changed first.)

- Write ICR[ICLCPLD] = 10h for Tx data fast frame.
- Write ICR[SNDICLC] = 1.

19. Both of the following operations are performed to change the LFAST masters Rx interface speed:

Note

(The slaves Tx interface speed mode should be changed first)

- Write ICR[ICLCPLD] = 80h to select Rx data fast frame.
- Write ICR[SNDICLC] = 1.

20. The ICLC transmission is confirmed by the occurrence of one of the following:

- ICR[SNDICLC] = 0
- TISR[TXICLCF] = 1.

21. Write SCR[TDR] = 1 to change the LFAST masters Tx interface speed.

22. Write SCR[RDR] = 1 to change the LFAST masters Rx interface speed.

23. Write ICR[ICLCPLD] = 00h to confirm the change in speed of the LFAST slave. This frame should be written after waiting for the expected delay in the start of the PLL and speed mode change delay at the LFAST slave.

24. Write ICR[SNDICLC] = 1

25. Write MCR[TXARBD] = 1, after some delay to ensure the step 24 ICLC frame is send but no other frame is sent to arbitration.

26. The LFAST slaves speed mode is confirmed by the occurrence of the following:

- If RIISR[ICPSF] = 1. The LFAST slave is in High Speed mode.
- If RIISR[ICPFF] = 1. The LFAST slave is in Low Speed mode.

27. If RIISR[ICPSF] = 1, then all frame arbitration is enabled by both of the following operations:

- Write ICR[ICLCSEQ] = 0.
- Write MCR[TXARBD] = 0.

28. If RIISR[ICPFF] = 1 modifying the LFAST slaves speed needs to be repeated by:

- Write SCR[TDR] = 0 to return the Tx interface to Low Speed mode.
- Write MCR[TXARBD] = 0, to enable frame transmission.
- The operations from step 14 need to be executed again.

45.8.1.2 LFAST slave interface startup procedure

1. After reset the SLCR and RCDCCR are programed according to the LVDS datasheet.
2. The PLLCR is programed with the configuration parameters for the PLL.
3. The LCR is programed with the configuration parameters for the LVDS.
4. Write MCR[MSEN] = 0. Then select LFAST modes by configuring MCR[CTSEN] and MCR[DATAEN].
5. Write MCR[DRFEN] = 1 to enable the LFAST.
6. The LR is enabled by writing MCR[RXEN] = 1. This negates the LR disable signal and LR's power down signal.
7. The LD enable signal needs to be asserted. This is done when one of the following are true:
 - When an ICLC frame with payload 31h is received the H/W will write RIISR[ICTEF] = 1 and MCR[TXEN] = 1.
 - MCRMCR[TXEN] = 1.
8. After a write to MCR[TXARBD] = 0, a ping frame will be sent on one of the following conditions:
 - When PICR[PNGAUTO] = 1 and an ICLC frame with payload 00h frame is received. H/W writes [ICRPF] = 1.
 - PICR[PNGREQ] = 1.

Speed mode change:

9. The PLL will start when one of the following conditions is met:

- When an ICLC frame with payload 02h is received. H/W writes RIISR[IPONF] = 1.
- Write PLLCR[SWPON] = 1.

10. The speed of the Tx interface is changed on one of the following conditions:

- When SCR[DRMD] = 1 and an ICLC frame with payload 80h is received. H/W writes RIISR[ICTFF] = 1 and SCR[TDR] = 1.
- SCR[TDR] = 1, when SCR[DRMD] = 0.

11. The speed of the Rx interface is changed on one of the following conditions:

- When SCR[DRMD] = 1 and an ICLC frame with payload 10h is received. H/W writes RIISR[ICRFF] = 1 and SCR[RDR] = 1.
- SCR[RDR] = 1, when SCR[DRMD] = 0.

12. A Ping Frame is sent on one of the following conditions:

- When PICR[PNGAUTO] = 1 and an ICLC frame with payload 00h is received. H/W writes RIISR[ICRPF] = 1.
- Write PICR[PNGREQ] = 1.
- Write PICR[PNGREQ] = .

45.8.2 Line Receiver

This section describes the Line Receiver.

45.8.2.1 Introduction

The LR detects the voltage swing on the differential pair and converts it to a CMOS logic level that feeds the adaptive auto correlation block. The received data is sampled using the best of the 8 (default) or 4 (alternative setting) possible sampling edges from the high speed clock or 4 phases using the low speed clock. The sampling edge is chosen by checking which of the 8/4 Correlators provide the maximum correlation. If more than one sampling edge provides the maximum then the state machine will choose the sampling edge based on a defined selection algorithm.

After the correct sampling edge is chosen, the remaining unused sampling edges are turned off. Once the header is received the length of the frame (payload size) and logical channel definition can be obtained. The logical channel definition will determine the data

type of the payload and therefore will determine the destination for the payload. Decoding of the ICLC commands from the payload is required in order to extract the interface control, rate mode and PLL commands.

Figure 45-48 shows the block diagram of Uplink Controller.

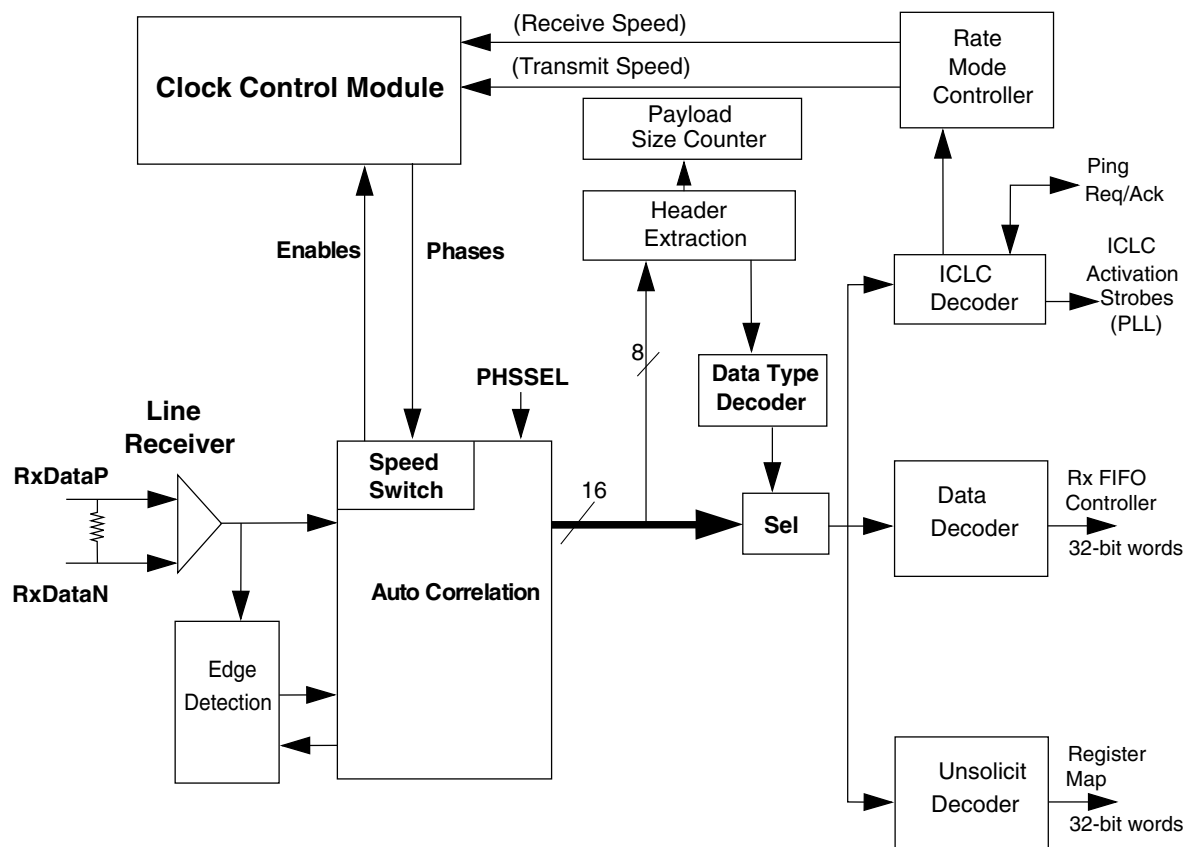


Figure 45-48. Top Level Receive Controller

45.8.2.2 Edge detection and auto-correlation

The edge detection and auto correlation are described together, as the mode setting of the auto-correlation block impacts largely on whether the edge detection circuitry is used or not. The edge detection will be performed first if required before auto-correlation.

45.8.2.2.1 Auto-Correlation modes

This section describes the Auto-Correlation modes.

45.8.2.2.1.1 Hunt Correlation mode

On reset the Receive Controller will always come up in Hunt Correlation mode. In this mode all the Correlators are enabled and the Receive Controller is always hunting for the synchronization pattern. The phase enables (either 8 or 4) to the external clock control module are always high. There is no edge detection for the first bit of the synchronization pattern. Hunt Correlation mode is considered the safest mode as the Receive Controller is always checking for the synchronization pattern, but it is the mode that consumes the most power. In Hunt Correlation mode the correlation is performed over all 16 bits of the synchronization pattern.

Figure 45-49 shows the block diagram of Hunt Correlation mode.

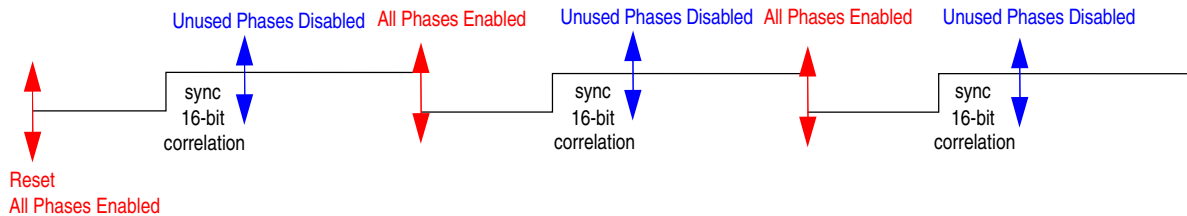


Figure 45-49. Hunt Correlation mode

45.8.2.2.2 Edge Detection

The edge detector is disabled in Hunt Correlation mode.

45.8.2.2.3 Auto correlation

The data transmission between the 2 devices LD and LR is asynchronous in nature. Hence the Receive Controller does not have the knowledge about the correct clock phase to be used for extracting the data. The task of the auto correlation (or synchronization) scheme is to estimate the best clock phase (8 or 4) for extraction of data as shown in the following figure.

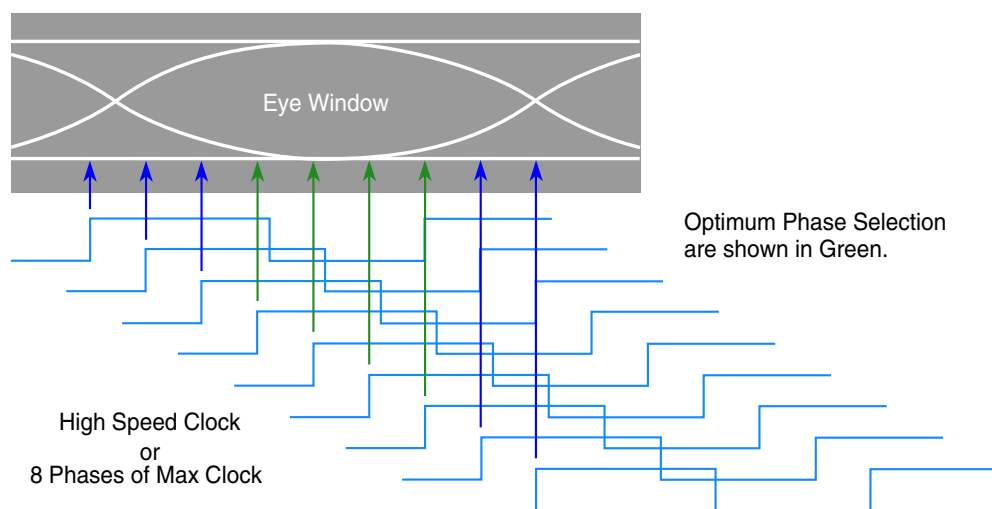


Figure 45-50. High Speed 8 Phase Selection

The objective of the auto correlation is to select the clock phase that occurs closest to the center of the eye diagram window. For 8 Phase clock alignment the worst case selection is when the clock edge is just after the LR output transition. The 8 clock phases will sample the correct value but the middle clock phases are the ideal selection. If one of the middle phases are selected then the minimum distance to the LR transition is 3 clock phases as shown in the following figure.

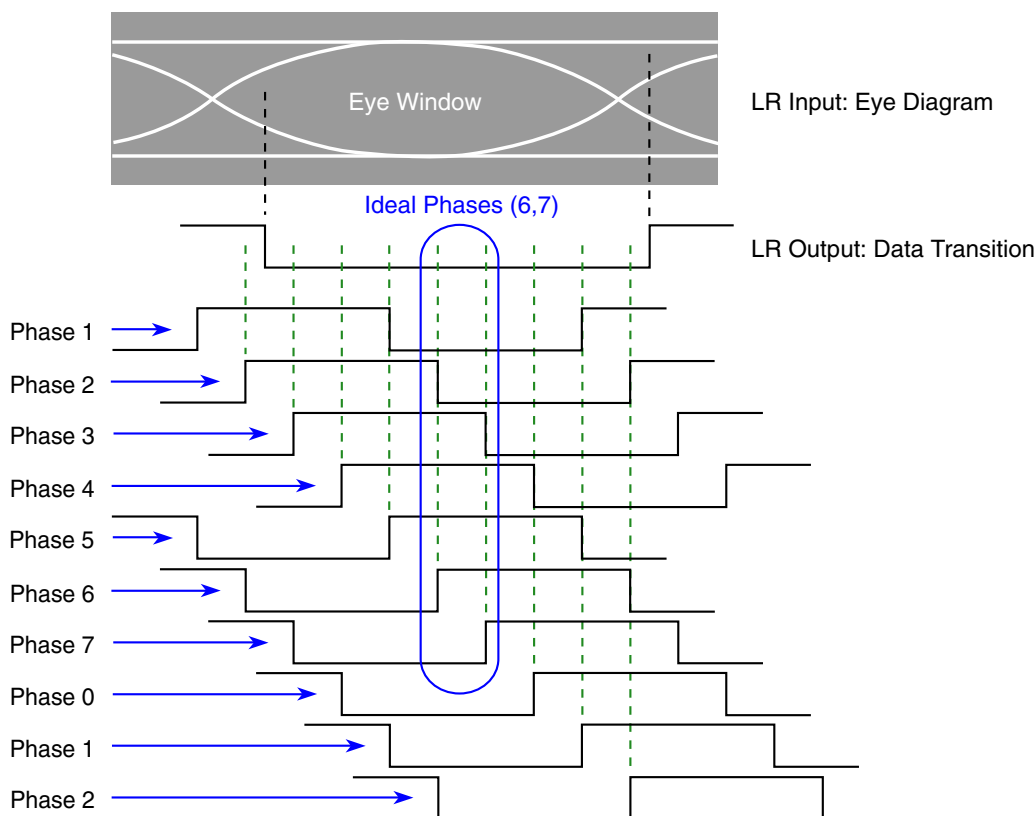


Figure 45-51. High Speed 8 Phase Clock Alignment Example

Each of the 8 high speed phases are 45 degrees separated. For 4 phases, whether high or low speed, the phases are 90 degrees separated but can have different phases enabled as shown in [Figure 45-52](#) and [Figure 45-53](#).

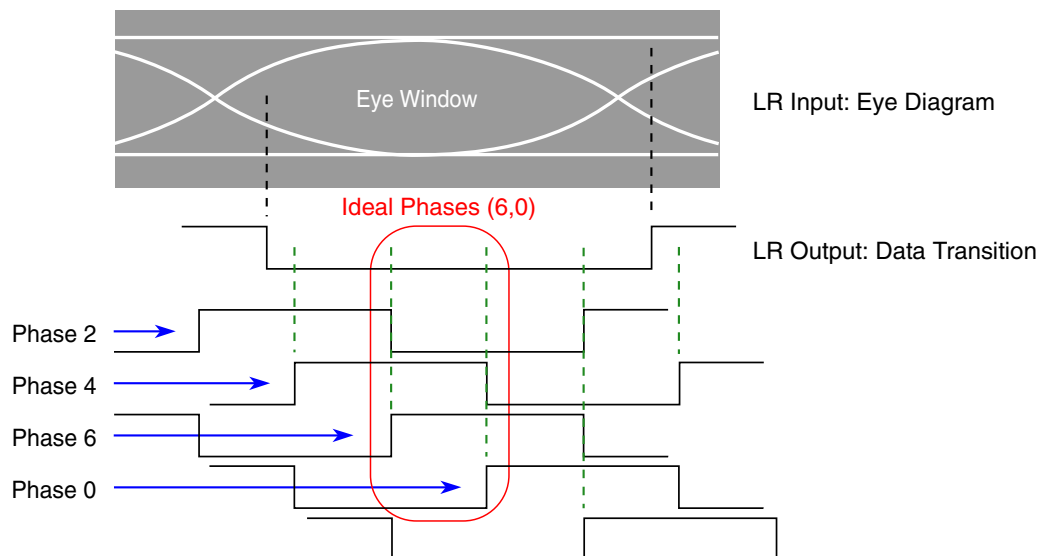


Figure 45-52. High Speed 4 Phase Clock Alignment Example

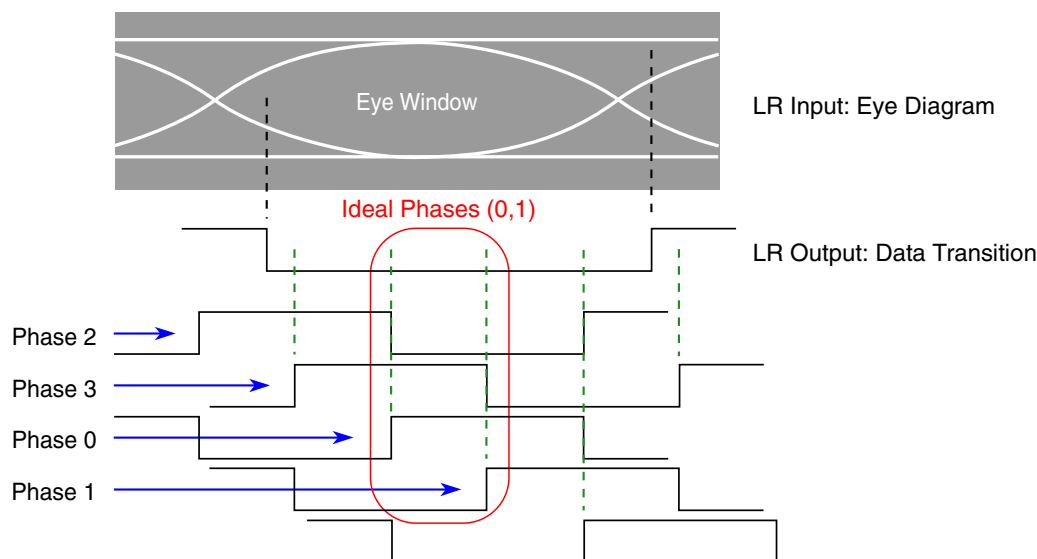


Figure 45-53. Low speed 4 phase clock alignment example

The auto correlation and selection of the correct clock phase starts after the edge detection finds a 0 to 1 transition. The high speed 8 or 4 phases from the PLL and the low speed 4 phases (generated inside the Clocking Module) are muxed inside the clocking module. The LFAST interface block will determine which phases are to be enabled and disabled. The only time the interface does not choose the phases is when the Clocking Module overrides the phase enables using COCR[SMPSEL].

The input data path can be sampled by different samplers depending on the configuration:

Functional description

- High speed 8-phases: Samplers 0,1,2,3,4,5,6,7
- High speed 4-phases: Samplers 0,2,4,6
- Low Speed 4-phases: Samplers 0,1,2,3

Each sampler block samples the data path with a different clock phase from the Clocking Module, and resamples it to a intermediate phase as shown in [Table 45-47](#), [Table 45-48](#), and [Table 45-49](#) before resampling it to Phase 0. The intermediate phase is used to make static timing between the initial phase and the final phase 0. The final phase, Phase 0 is chosen so all the clock trees after the sampler are clocked by the same clock.

Table 45-47. High speed 8 phase selection - sampling procedure

Samplers	InitialSample	Intermediate Sample	Final Sample
0	Phase 0	Phase 0	Phase 0
1	Phase 1	Phase 0	Phase 0
2	Phase 2	Phase 0	Phase 0
3	Phase 3	Phase 0	Phase 0
4	Phase 4	Phase 2	Phase 0
5	Phase 5	Phase 2	Phase 0
6	Phase 6	Phase 4	Phase 0
7	Phase 7	Phase 4	Phase 0

Table 45-48. High speed 4 phase selection - sampling procedure

Samplers	InitialSample Phase	Intermediate Sample Phase	Final Sample Phase
0	Phase 0	Phase 0	Phase 0
1	Disabled	Disabled	Disabled
2	Phase 2	Phase 0	Phase 0
3	Disabled	Disabled	Disabled
4	Phase 4	Phase 2	Phase 0
5	Disabled	Disabled	Disabled
6	Phase 6	Phase 4	Phase 0
7	Disabled	Disabled	Disabled

For High speed 4 Phase selection, See [Table 45-48](#), Samplers 1, 3, 5, 7 are disabled. In the Phase Select algorithm Phase 1 is mapped to Phase0, Phase 3 is mapped to Phase 2, Phase 5 is mapped to Phase 4, Phase 7 is mapped to Phase 6 so it simplifies the algorithm and allows the algorithm to be the same independent of 4 or 8 Phases in high speed.

Table 45-49. Low speed 4 phase selection - sampling procedure

Samplers	InitialSample Phase	Intermediate Sample Phase	Final Sample Phase
0	Phase 0	Phase 0	Phase 0

Table continues on the next page...

Table 45-49. Low speed 4 phase selection - sampling procedure (continued)

Samplers	InitialSample Phase	Intermediate Sample Phase	Final Sample Phase
1	Phase 1	Phase 0	Phase 0
2	Phase 2	Phase 0	Phase 0
3	Phase 3	Phase 0	Phase 0
4	Disabled	Disabled	Disabled
5	Disabled	Disabled	Disabled
6	Disabled	Disabled	Disabled
7	Disabled	Disabled	Disabled

For low speed 4 Phase selection, See [Table 45-49](#), Samplers 4, 5, 6, 7 are disabled. The first four Samplers (0,1,2,3) of the low 4 phase speed selection algorithm are the same as the four samplers required for the high 8 phase speed. Therefore the same architecture can be used for both high speed and low speed with the other four data paths disabled for low speed.

45.8.2.2.4 Sampler block and phase enable and disable

There are 8 data sampler paths in total inside the auto correlation block, each data sampler has 3 sampling registers with the possibility of each register being clocked by a different phase (in example, Sampler 5 for high speed can have Phases 5, 2 and 0). Therefore, after the correct data sampler has been selected for either high or low speed, the block can turn off all the sampler paths except for one, therefore 3 out of 24 registers will remain enabled while the others are disabled, as shown in the following figure.

After correlation and the correct data sampler has been selected all the other data samplers whose initial phase does not match the select phase are disabled. The selected sampler will then keep the required phases needed enabled, this can be max 3 phases (for example, Sampler 5 has Phases 5, 2, 0) or min 1 phase (in example below, Sampler 0 has all Phase 0 content).

The end result is that the Rx Controller can disable the required number of unused sampler registers and disable any unnecessary phases from the Clocking Module by deasserting the respective phase enables.

Functional description

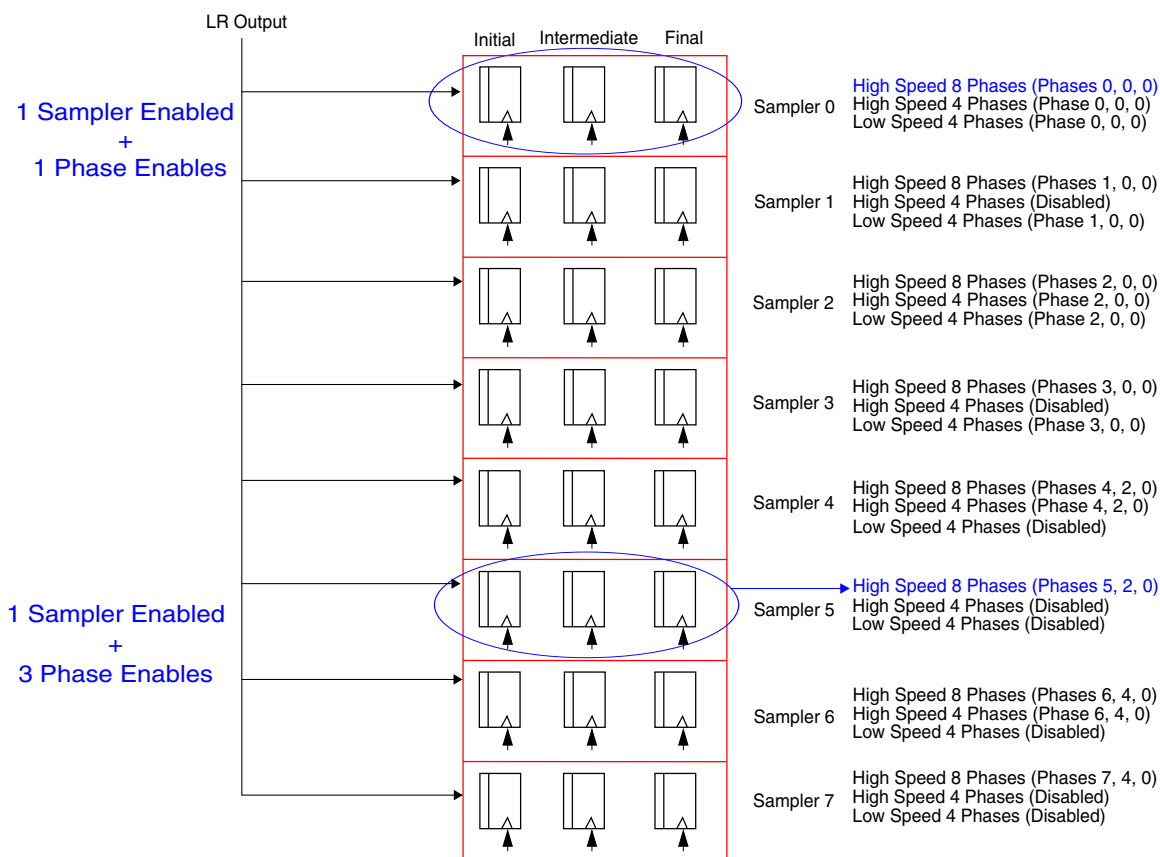


Figure 45-54. 8 Samplers, each Sampler has 3 Registers

In order to achieve the sampler and phase enable requirements each Sampler block will require the logic as shown in the following figure.

The Clock Gating Element resides inside the Clocking Module block. The interface will provide the enables to the Clocking Module and the Clocking Module will in return provide the individual clocks to the sampling registers.

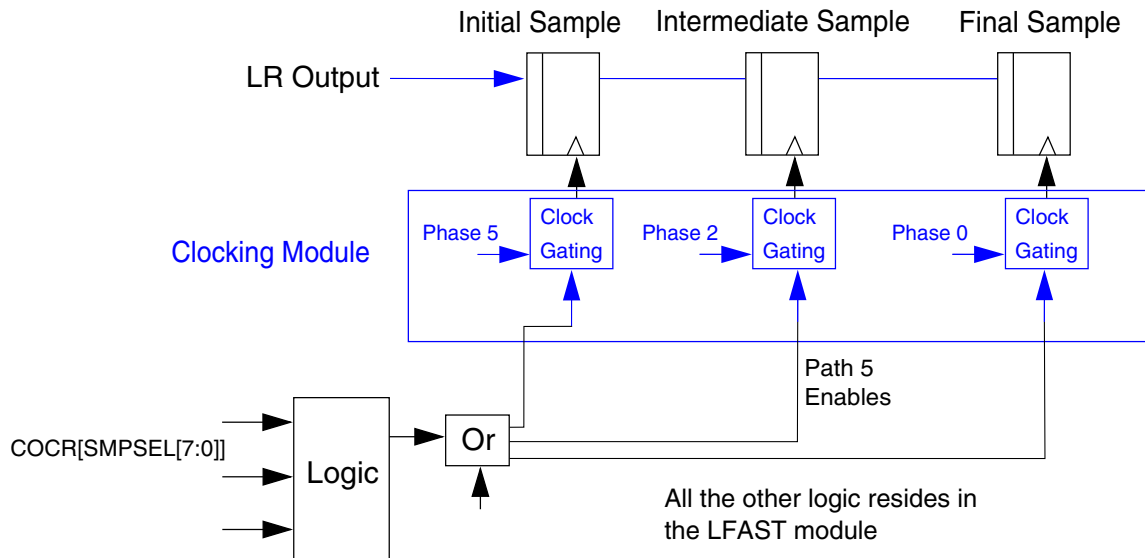


Figure 45-55. Sampler 5 Logic

45.8.2.3 Header and Payload Extraction

Once the Phases/Samplers are chosen after Synchronization and Correlation the next part of the flow is the header extraction. The Receive Controller will output a 16 bit word, bit shifted every Phase 0 clock as shown in the [Figure 45-56](#).

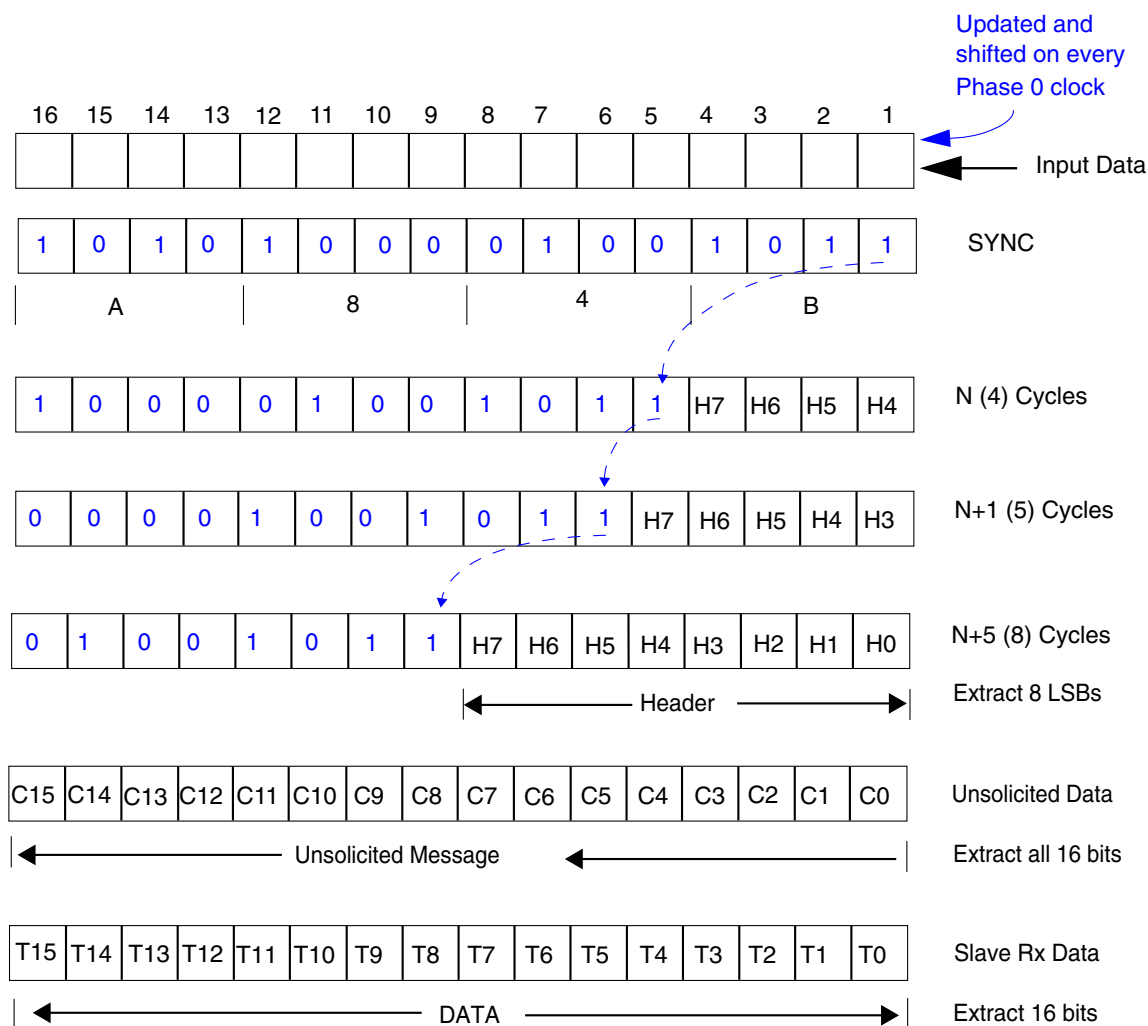


Figure 45-56. Extracting Header and Payload from the 16-bit output from Auto correlation.

45.8.3 Transmit Controller

This section describes the Transmit Controller.

45.8.3.1 Introduction

The Transmit Controller uses Rx information, status information and error control data and codes them into the appropriate frame structure for transmission to the LD. This includes the synchronization and header, in preparation for the LD, which transmits the frame to peer LFAST IC at either low or high speed. The Transmit Controller creates a frame structure that is converted to a serial format for the LD.

An arbitration block will determine which message has higher priority data, unsolicited, ICLC, CTS, ping, and so on. The data frames are extracted from the FIFO controller, the unsolicited message from the register block, the CTS messages from the Receive Controller and the ping response from the register block.

The Tx interface has two speed modes:

- Low Speed
- Fast Speed

The Transmit Controller consists of the following blocks as shown in the following figure.

- Arbitrator
- Framer
- Request Clock Control

The arbitrator will grant access to the framer from the request that has the highest priority. The framer block will extract the payload data from the granted request source and build the frame (adding synchronization or header if required). The frame is fed into a PISO (Parallel In Serial Out) and eventually sent to the LD.

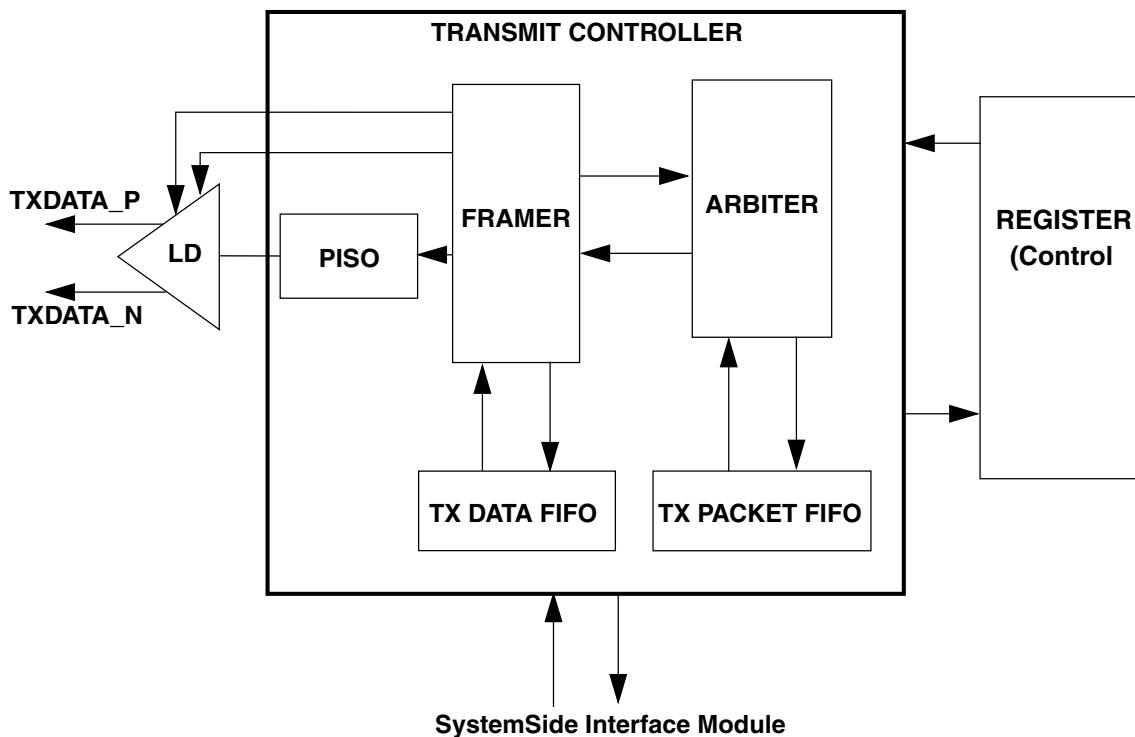


Figure 45-57. Transmit Controller Connections

45.8.3.2 Arbitration

The arbitration block prioritizes between requests from multiple sources like

- S/W programmable registers
- System side interface FIFO
- Rx interface controller

The level of priority for each request is shown in [Table 45-50](#)

Table 45-50. Priority Levels for the Transmit Controller

Request	LFAST MasterPriority	LFAST SlavePriority	Triggering Conditions (at least one of the listed)
LFAST interface enable	1	1	• LFAST interface enable is asserted • LFAST interface enable is negated
Tx Interface disabled	2	2	• MCR[DRFEN] = 0 • MCR[TXEN] = 0 • MCR[DRFRST] = 1 • MCR[TXARBD] = 1 • LFAST Slave: ICLC frame with payload for "Disable Rx interface" received
Tx Interface speedmode change	3	3	• SCR[TDR] is modified • LFAST Slave: ICLC frame with payload for changing Rx interface speed received
Loopback frame in Loopback mode	4	4	• TMCR[LPON] = 1Dig • RF Slave: ICLC frame with payload for loopback mode enable received Note: Valid only for following TMCR[LPMOD] settings: LPMOD[2:0] = 011b LPMOD[2:0] = 100b
ICLC frame request	5	-	• LFAST Master: ICR[SNDICLC] = 1 • LFAST Master: ICR[ICLCSEQ] = 1
Ping response request	-	5	• LFAST Slave: ICLC frame with payload for ping request received and PICR[PNGAUTO] = 1 • LFAST Slave: PICR[PNGREQ] = 1

Table continues on the next page...

Table 45-50. Priority Levels for the Transmit Controller (continued)

Request	LFAST MasterPriority	LFAST SlavePriority	Triggering Conditions (at least one of the listed)
Unsolicited frame request	6	6	- Last Frame with CTS = 1 received from the peer LFAST device - UNSTCR[USNDRQ] = 1
Data frame from Tx FIFO	7	7	- Tx FIFO contains one or more frames - Last Frame with CTS = 1 received from the peer LFAST device - MCR[DATAEN] = 1
CTSFrame	8	8	- MCR[CTSEN] = 1 - Rx FIFO reaches High/Low threshold, defined by TISR and no frame pending
Clock mode test	9	9	- TMCR[CLKTST] = 0 - LFAST Slave: ICLC frame with payload for clock test mode enable received

Once the arbitrator has granted access to a request, the Framer will commence building the frame. Any new requests for frame or data rate change will not be granted access until the framer has finished transmitting the current frame. If a data rate change request for the Tx interface is received while the Transmit controller is in the middle of sending a frame then the rate change request to the Clocking Module will be delayed. This allows for the frame to be completed before the change in speed mode. This prevents any speed mode change during the transmission of a frame. Once the speed mode request is sent to the Clocking Module by the Tx Interface Controller, it will then not allow any new requests to be processed for a specific time period defined by the bit field RCDCR[DRCNT].

45.8.3.3 Line Driver digital connections

45.8.3.3.1 Line Driver states

The LD has the following states:

- Shutdown

The LD enters the Shutdown state when the LD powerdown signal and the output buffer enable is high. In this state, the LD regulator is not supplying power and the LD outputs are connected to ground. Once LD powerdown signal is negated, a settling time is required before the LD may be used for communication. The settling time is defined by the SLCR[LWKCNT] and SLCR[HWKCNT] bitfields.

- Sleep

The LD enters in sleep state when the signal is asserted. In this state, the LD is enabled, but held in a power-saving state. Sleep mode may be used during inter-frame gaps that are long compared to the frame durations but not long enough to allow the interface(s) or high-speed clock generators to be powered down completely. In the sleep state the LD outputs are connected to V_{cm} (common mode voltage). To exit from Sleep mode the LD sleep signal is negated. The LD is required to transmit a logic 0 level on the interface for a pre-defined minimum time. The minimum time is the summation of settling time of LD after negation of LD sleep signal and the wakeup time of the LR on the other side. This delay is programed in the SLCR[HSCNT] and SLCR[LSCNT] bitfields.

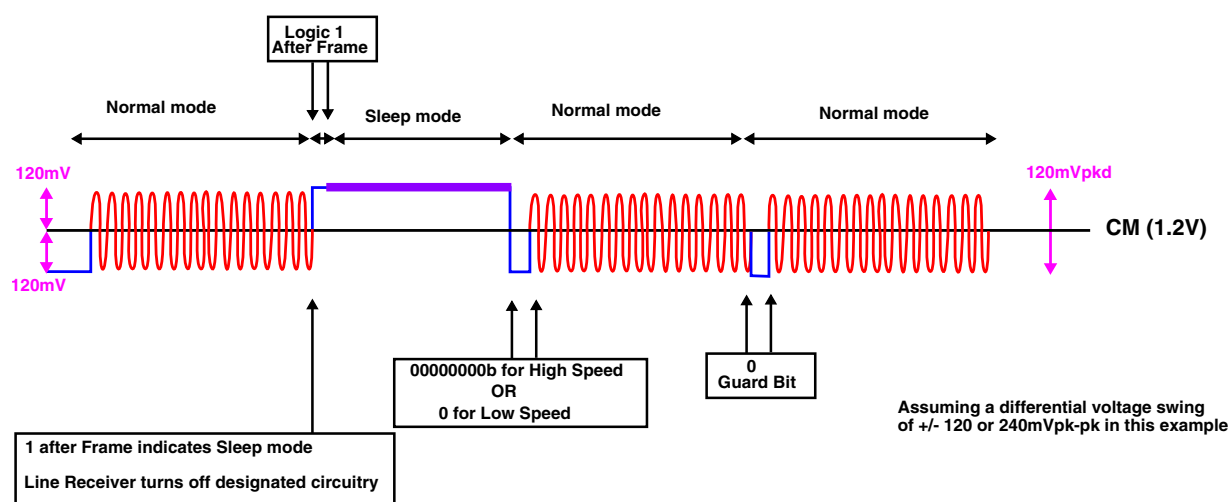


Figure 45-58. Example of Sleep mode

As shown in [Figure 45-58](#) before every normal mode burst on the txdatap line there is one guard bit period (minimum inter-frame gap) where a logic 0 will be transmitted.

- Normal

In the Normal state the LD is primed for transmission. The LD shall drive the interface as dictated by the input data bit that is supplied by a shift register under the control of a finite state machine. If the LD is not enabled, and the framer is activated,

and has a frame to transmit, an interrupt (TISR[TXIEF]) will be generated. The LD moves back to the shutdown state when the LFAST master sends an ICLC Rx data off command.

Table 45-51. Line Driver States

LD State	LFAST Master Triggers	LFAST Slave Triggers
Shutdown	• Programing LVDS[SWOFFLD] and LVDS[SWONLD] • Programing MCR[TXEN] and MCR[DRFEN]	• LFAST interface enable negation/assertion • Programing LVDS[SWOFFLD] and LVDS[SWONLD] • ICLC command from LFAST master • Programing MCR[TXEN] and MCR[DRFEN]
Sleep	• Programing LVDS[SWSLPLD] and LVDS[SWWKLD]	• No pending frame for transmission • Programing LVDS[SWSLPLD] and LVDS[SWWKLD]
Normal	Absence of above mentioned conditions	Absence of above mentioned conditions

45.8.4 CTS mode support

LFAST module supports flow control methods. The CTS (Clear to send) is a protocol defined method to ensure no loss of frame, due to Rx FIFO unavailability. The optimal use of bandwidth of LFAST, without losing frames, is ensured by proper programing of the Rx FIFO Lower threshold (RFCR[CTSMN]) and Higher threshold (RFCR[CTSMX]). The CTS is supported by both the LFAST Master and Slave.

CTS Transmission:

The LFAST device sends CTS information with each frame to the peer LFAST device.

If the CTS mode is enabled by write $MCR[CTSEN] = 1$ then:

- Sends CTS bit as 0 in LFAST frame (bit[0] of Header field) whenever the Rx FIFO reaches the higher threshold defined by the bitfield TISR[CTSMX]. This indicates that the LFAST device doesn't want the peer device to send data. The CTS bit of all frame are sent 0 untill the Rx FIFO pointer reaches the lower threshold, as described below.

- Sends CTS bit as 1 in LFAST frame (bit[0] of Header field) whenever the Rx FIFO reaches the Lower threshold defined by the bitfield TISR[CTSMN]. This indicates that the LFAST device is ready to receive data from the peer device. The CTS bit of all frame are sent 1 untill the Rx FIFO pointer reaches the Higher threshold, as described above.
- The CTS bit can be sent through any type of frame (data, unsolicited or ICLC). When there are no frames available in LFAST to send to the peer device, a CTS frame is sent. In this frame, bit[4-1] of Header field are sent as 0011b and 8-bit payload of 0.

If the CTS mode is not enabled (for example, bit MCR[CTSEN] = 0) then:

- All the frame sent will have CTS bit as 1 in LFAST frame (bit[0] of Header field).

CTS Reception:

The reception of a LFAST frame with CTS bit 0 indicates that the peer device is not ready to receive data frames. The transmission of all pending data and unsolicited frames are postponed untill a frame with CTS bit 1 is received.

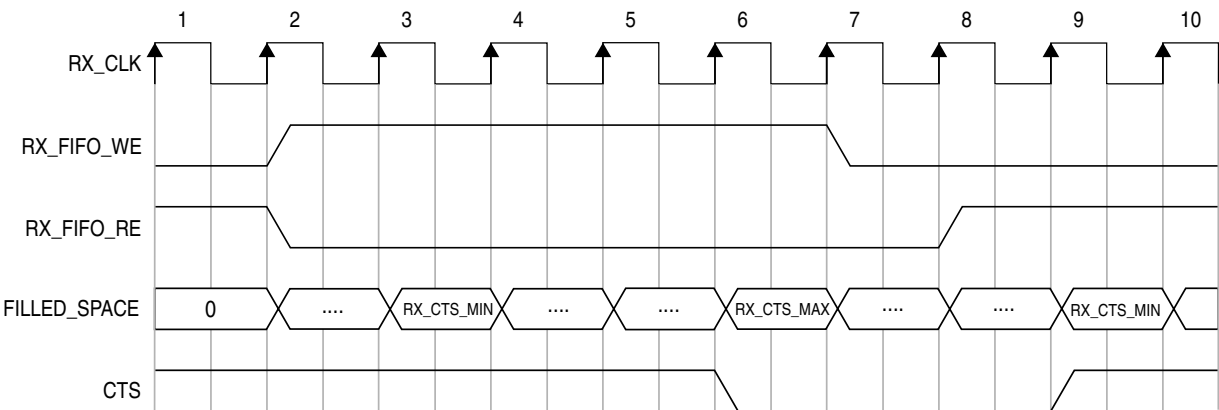


Figure 45-59. CTS generation

45.8.5 Frames supported

Table 45-52 provides the frames supported and their permissible payload sizes

Table 45-52. Frames supported by LFAST interfaces

Frame Type	LCT Code	Supported by 1. LFAST Master Tx2. LFAST Slave Rx	Supported by1. LFAST Slave Tx2. LFAST Master Rx	Payload Size (bits)
Data Frame	0100b – 1011b	YES	YES	32, 64, 96, 128, 256, 288
Unsolicited Frame	0001b	YES	YES	8, 32, 64, 96, 128, 256 and 288
ICLC Frame	0000b	YES	NO ¹	8
CTS Frame	0011b	YES	YES	8
Reserved	All others	—	—	—

1. Except the ICLC PING response frame.

If logical channel type (LCT) code other than the above mentioned are received then the interrupt flag $RISR[RXLCEF] = 1$. If the payload size received does not match any value in [Table 45-52](#) then the flag $RISR[RXSZF] = 1$. In both these error conditions the received frame is ignored. The S/W may have to take the necessary steps to recover from such an error.

45.8.6 Frame flow

Following sections describe Data Frame Flow, ICLC Flow and Rx Unsolicited Data Flow.

45.8.6.1 Data flow

LFAST supports the transfer of data between master and slave LFAST devices.

45.8.6.1.1 Data transmit

System Side Module is the initiator of all the Tx data frame to LFAST peer device. Data frame transmit is triggered whenever the Tx Data FIFO has at least one valid frame. The $MCR[DATAEN]$, $MCR[DRFEN]$ and $MCR[TXEN]$ bits, should be set to enable the transfer of Tx data.

If $MCR[DATAEN] = 0$, then the valid frames in the in Tx data FIFO will be ignored for transmission. The Payload Size and channel type of each frame is specified by the System Side Module interface during transfer of the frame to the LFAST interface. The frame header is stored in the Tx packet FIFO and the payload in the Tx data FIFO. Whenever the Tx FIFO has at least one frame then a data transmit request is made to the Tx arbiter of the LFAST. Tx block arbitrates the data transmit request and

schedules it depending on the priority of all the pending transmit requests. When data request is scheduled by Tx block, the required data is fetched from Tx data FIFO. The number of frames present in the Tx FIFO for transmission is indicated by the bitfield DFSR[TXFCNT].

45.8.6.1.1.1 Programing model for Tx data transmit

1. Program MCR[DATAEN] = 1, MCR[TXEN] = 1 and MCR[DRFEN] = 1 to enable the Tx path of LFAST device
2. Select the desired clock rate at which data should be transmitted, using ICLC transfers and appropriate programing of SCR[TDR] bits.
3. Frame present in TX FIFO (Data and Packet FIFO) are sent.
4. TISR[TXDTE] = 1 after each frame transfer
5. CTS is set depending on the push/pull mode setting defined by MCR[CTSEN].

45.8.6.1.2 Data receive

LFAST master and slave supports reception of data frame. The received frame is determined to be of data frame type by decoding channel type field of the header present in the received frame. The MCR[DATAEN], MCR[RXEN] and MCR[DRFEN] bits should be set to enable the data frame reception. When MCR[DATAEN] = 0 the received data frames will be ignored and will not be placed in the Rx data FIFO.

The Rx data frames received by Rx block are stored in the Rx FIFO. Whenever the frame is received in the Rx FIFO the System Side Module is indicated by assertion of LFAST Rx FIFO ready signal. The frame size and the Channel type is passed to the LFAST. The frame boundaries are indicated by start of frame and end of frame signals. The number of unread frames in the Rx Data FIFO are indicated by DFSR[RXFCNT]. When Rx DATA FIFO is full and cannot accommodate current frame completely, then the remaining data of the Rx frame is discarded. In this case, UNSRSR[RXOF] = 1 and an interrupt is generated if the RIER[RXOFIE] = 1.

45.8.6.2 Unsolicited flow

45.8.6.2.1 Unsolicited frame transmit flow

The S/W is the initiator for unsolicited frames to the LFAST peer device. The unsolicited frame header and payload is programmed into the Unsolicited Data and Control registers. Once the Payload is programmed UNSTCR[USNDRQ] is set, generating a request for unsolicited frame transfer to the Tx arbiter.

45.8.6.2.1.1 Programing model for unsolicited frame transmit

1. Program MCR[TXEN] = 1 and MCR[DRFEN] = 1 in the Mode Configuration Register (MCR) to enable the Tx path
2. Select the desired clock rate at which data should be transmitted, using ICLC transfers and appropriate programming of SCR[TDR].
3. Read the UNSTCR[USNDRQ].
 - If UNSTCR[USNDRQ] = 1 then wait for either of the following:
 - UNSTCR[USNDRQ] = 0
 - TISR[TXUNSF] = 1
 - If UNSTCR[USNDRQ] = 0 then:
 - Program the unsolicited payload in UNSTDR0–UNSTDR8, and can be written up to a payload of frame of a maximum of 288 bits.
 - Program the unsolicited frame header in UNSTCR[UNSHDR].
 - Program UNSTCR[USNDRQ] = 1.
4. UNSTCR[USNDRQ] is cleared by the Tx Block after the frame transfer.
5. TISR[TXUNSF] = 1 when the frame is transmitted.
6. CTS is set depending on the Push-Pull mode setting defined by MCR[CTSEN].

45.8.6.2.2 Unsolicited frame receive flow

Whenever an Unsolicited frame is received from the peer device, the following steps are performed:

If UNSRSR[URXDV] = 0 then:

1. The payload size field of the header is first saved into the UNSRSR[URPCNT].
2. The payload is stored in the UNSTDR0–UNSTDR8.

3. UNSRSR[URXDV] = 1 and the RISR[RXUNSF] = 1 indicating the successful reception of the frame.

If UNSRSR[URXDV] = 1 then:

1. The current unsolicited frame is ignored.
2. RISR[RXUOF] = 1.

Typical steps by the processor after RISR[RXUNSF] = 1 is as follows:

1. The processor reads UNSRSR to get the payload size of the frame.
2. It then reads the complete frame by reading UNSRDR8–UNSRDR0 for the payload size as received in the frame.
3. Then it clears the interrupt (write RISR[RXUNSF] = 1) and also the UNSRSR[URXDV] bit.

45.8.6.3 ICLC flow

The ICLC (Interface Control Logical Channel) is a separate logical channel type, which is mainly meant for implementing the data rate change in the LFAST interface and initiating the test modes.

45.8.6.3.1 ICLC data transmit flow

Whenever the processor intends to transfer an ICLC frame to the LFAST slave then the following steps are to be followed.

1. Write the ICLC frame payload in the ICR[ICLCPLD] (see [Table 45-53](#) for payloads as defined by the standard).
2. Program ICR[SNDICLC] = 1 to initiate the ICLC frame transfer. This bit clears itself when the ICLC frame has been transmitted. The processor needs to poll this bit to make sure it is cleared before setting this bit again (even when ICR[ICLCSEQ] = 1).
3. TISR[TXICLCF] = 1 after the transfer of the ICLC frame.
4. CTS is set depending on the Push-Pull mode setting defined by MCR[CTSEN].

To determine whether the ICLC frame has been sent or not, either the ICR[SNDICLC] bit can be polled, or wait for TISR[TXICLCF] = 1.

Table 45-53. Supported ICLC Payloads

ICLC Code(hex)	ICLC Function
00	Ping Request from LFAST Master to LFAST Slave
01	Reserved
02	Start PLLIn preparation for High Speed mode
04	Stop PLL Fallback to low speed mode
08	Select TxData Slow (LFAST Slave Rx Interface)
10	Select TxData Fast (LFAST Slave Rx Interface)
20	Select RxData Slow (LFAST Slave Tx Interface)
40	Not Supported
80	Select RxData Fast (LFAST Slave Tx Interface)
31	Enable RxData Interface (LFAST Slave Tx Interface)
32	Disable RxData Interface (LFAST Slave Tx Interface)
34	Clock Test mode Send 101010... continuously using the currently configured clock rate (slow, fast);
FF	Turn payload loopback on
38	Turn Test mode (Loopback and Clock Test) off

Programing model for ICLC frame transmit:

1. Set the ICR[ICLCSEQ] bit (optional)
2. Write the ICLC frame payload in the ICR[ICLCPLD], corresponding to the desired Tx/Rx data rate change, as described in the [Table 45-53](#).
3. Write ICR[SNDICLC] = 1 to initiate the ICLC frame transfer.
4. LFAST master will schedule the ICLC transfer. Data present in ICR[ICLCPLD] is transmitted to the LFAST slave at the old Tx data rate. LFAST slave will configure its Tx/Rx interface data rate accordingly.
5. Reset the ICR[ICLCSEQ] to allow the scheduling of other types of frames, which will be sent at the new data rate.

When ICR[ICLCSEQ] = 0, though the ICLC frame transfer still has highest arbitration priority, other frames may be scheduled if no valid ICLC frame request exist.

Note

ICLC frame transmit will not trigger the internal data rate change of LFAST Master Tx/Rx interface. LFAST master Tx/Rx interface data rate will only be changed, when corresponding SCR[TDR] and SCR[RDR] bits are appropriately configured.

Processor requires to take care that no data frames are to be transmitted while data rate change is happening or ICLC is transmitting frames by setting ICR[ICLCSEQ] bit in [Table 45-53](#).

45.8.6.3.2 ICLC data receive flow

When the bits 4 to 1 of a receive frame are 0000b it indicates that the payload is an ICLC. ICLC payloads are always 8 bits.

The reception of an ICLC frame is indicated by an interrupt to the system. The ICLC status register RISR indicates the type of the ICLC frame received. The Rx block will decode and generate the appropriate signals. An invalid ICLC code besides the ones listed in [Table 45-53](#) will cause RISR[RXICF] = 1.

45.8.6.3.2.1 Ping request ICLC

The LFAST slaves ICLC decoder will decode the ping request and set RIISR[ICPRF]. The S/W can also write to PICR[PNGREQ] to indicate to the Tx block that a ping response frame needs to be transmitted. The PICR[PNGAUTO] indicates whether Tx block can respond automatically to the request by the LFAST master ICLC Ping Request frame. The Tx block will arbitrate the ping response frame request and send a ping frame with ping data defined by bitfield PICR[PNGPLYD]. Once the ping response frame has been sent the Tx block will set TISR[TXPNGF] and clear PICR[PNGREQ], if set.

45.8.6.3.2.2 LFAST Slaves RxData Interface Slow/Fast ICLC

The LFAST Slaves Rx Interface has two speed modes supported: slow (Low speed) and fast (High speed). The ICLC command will write RIISR[ICRSF] = 1 (Low speed) or RIISR[ICRFF] = 1 (High speed).

45.8.6.3.2.3 LFAST Slaves TxData Interface Slow/Fast ICLC

The LFAST slaves Tx Interface Controller has two speed modes: slow (Low speed), and fast (High speed). The ICLC command will write RIISR[ICTSF] = 1 (Low speed) or RIISR[ICTFF] = 1 (High speed).

45.8.6.3.2.4 Enable/Disable LFAST Slaves TxData Interface ICLC

The ICLC commands, Enable RxData Interface and Disable RxData Interface are decoded and the appropriate enable/disable signal sent to the Tx block Controller. These ICLC command writes RIISR[ICTEF] = 1 and RIISR[ICTDF] = 1. The ICLC Tx enable command will write MCR[TXEN] = 1.

No frames can be sent on the LFAST Slave Tx interface until the either LFAST master has enabled it via an ICLC frame or S/W programs MCR[TXEN] = 1.

45.8.6.3.2.5 Clock Test mode ICLC

This ICLC command is decoded, and then is used to write TMCR[CLKTST] = 1. This indicates to the Tx interface to output an alternating pattern of 1 and 0 at the currently configured clock rate. The Rx interface generates RIISR[ICCTF] = 1 on reception of this ICLC command.

The exit from this mode happens on reception of Test mode off ICLC command.

45.8.6.3.2.6 Loopback payload on ICLC

This ICLC command is decoded and TMCR[LPON] = 1 to indicate to the Tx Block that the received payload is to be sent back. The exit from this mode happens on reception of Test mode off ICLC command. The Rx interface causes RIISR[ICLPF] = 1 on reception of this ICLC command.

45.8.6.3.2.7 Test mode off ICLC

This ICLC command is decoded and TMCR[LPON] and TMCR[CLKTST] are cleared to indicate to the Tx block to exit from loopback mode or clock test mode.

Another option is for the payload loopback option to remain enabled until negated on the toggling of LFAST interface enable.

45.8.6.3.2.8 Ping response ICLC

The LFAST master considers any ICLC frame payload as ping response data. The LFAST masters Rx block will store the received ping data in PISR[RXPNGD] and match it with PICR[PNGPYLD]. RIISR[ICPSF] = 1 if the Ping received matches with the one stored in PICR[PNGPYLD], or RIISR[ICPFF] = 1.

45.8.6.4 CTS flow

45.8.6.4.1 CTS Tx flow

The CTS frame, if enabled by the S/W, indicates the status of the Receive Data FIFO. CTS frame is triggered whenever Rx Data FIFO reaches either the High/Low threshold and no data, ICLC or unsolicited frame is ready for transmission.

Programing model for CTS frame transmit:

- Program RFCR[CTSMX] and RFCR[CTSMN].
- Enable the CTS frame Transmission by programming MCR[CTSEN] = 1.
- Program MCR[TXEN] = 1 and MCR[DRFEN] = 1 to enable the Tx path.
- The CTS bit sent with a valid frame (for example, data, ICLC and unsolicited) is 1 until the Rx Data FIFO reaches High Threshold.
- When the Rx Data FIFO reaches higher threshold and no valid frame (data, ICLC and unsolicited) is pending for transmission, then the CTS frame is triggered, with CTS bit of header = 0 and the status bit RISR[RXMXF] = 1.
- When the Rx Data FIFO reaches Low Threshold, due to system side reads and no valid frame (data, ICLC and unsolicited) is pending for transmission, then the CTS frame is triggered, with CTS bit of header = 1, and status bit RISR[RXMNF] = 1.

45.8.6.4.2 CTS Rx flow

Whenever a CTS frame header or any other frame with valid header is received with CTS bit = 0, then RISR[RXCTSF] = 1. The Tx Interface arbitration for unsolicited frame and data frame transmit request is turned off on reception of frame with CTS bit 0. The arbitration for unsolicited frame and data frame is enabled on reception of frame with CTS bit 1. Frame with LCT type CTS are not stored in the LFAST.

45.8.7 Test and Debug Support

The test and debug interface helps to debug the LFAST module. These signals can be brought out for ease of validation.

45.8.7.1 Loopback Test mode

The loopback function allows to verify the correct operation of the physical interface and the basic checks for the LFAST module without a peer device.

There are certain prerequisites before entering the Loopback mode:

- LD and LR should be turned on.
- Tx and Rx mode should be enabled.
- Interrupts needed for S/W should be enabled.
- Both Tx and Rx interface should be in same speed mode.
- For Automatic Test mode TMCR[LPFRMTH] should be programmed.

The LFAST module supports four loopback modes, defined by TMCR[LPMOD].

Table 45-54. Loopback modes

LPMOD[2:0]	Mode Selected
000	Rx Loopback (default)
001	Rx LVDS Loopback
010	Tx Loopback without Automatic frame generation
011	Tx Loopback with Automatic frame generation
100	Tx LVDS Loopback (external) with Automatic frame generation

The TMCR[LPON] bit controls the state of the loopback function, and the default setting is 0 (off). This can be written by S/W, or in the case of the LFAST slave, the TMCR[LPON] bit can be modified by the Rx interface controller on decoding of a valid ICLC loopback on or Test mode off commands. The LPMOD should not be changed when the LPON bit is set.

In all loopback modes, except Tx loopback without automatic frame generation, the decoding of frame and error flags is stopped. Only decoding of following is done:

- ICLC Turn Test mode Off frame.

- CTS bit of all frames.
- Automatic Loopback frame decoding (only in Automatic frame generation mode).

45.8.7.1.1 Rx Loopback mode

For Rx loopback mode the output of the LR is passed to the LD with manipulation via the Rx and Tx Interface controllers. CTS bit (Bit 0) of the header is guaranteed to be asserted when the incoming data from the other device is looped back. In Rx Loopback mode the Rx Interface controller operates in normal mode, decoding the frames (header, payloads). This allows the LFAST Master to control the Loopback mode on and off by sending ICLC commands.

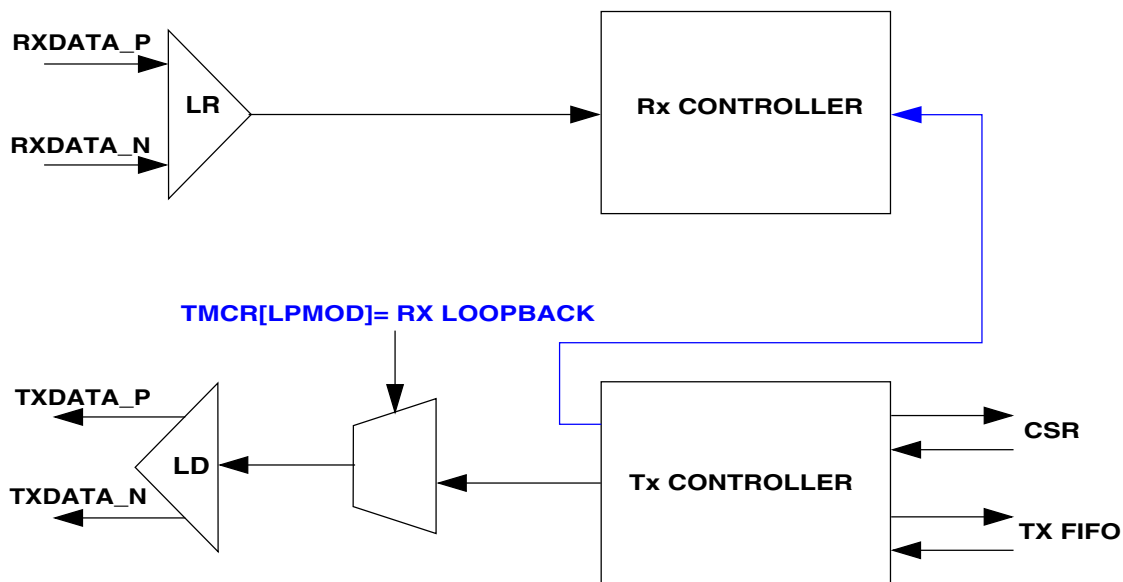


Figure 45-60. Rx LoopBack mode

Entry to Rx Loopback mode:

1. S/W programs $\text{TMCR}[\text{LPMOD}] = 000\text{b}$.
2. Loopback can be turned on by either of the following methods:
 - S/W programs $\text{TMCR}[\text{LPON}] = 1$.
 - For Slave Only: - Reception of ICLC Frame with Payload FFh (Loopback mode on), from LFAST Master

Exit from Rx Loopback mode can be done by any of the following methods:

- S/W programs $\text{TMCR}[\text{LPON}] = 0$
- For LFAST Slave: - Transmission of ICLC Frame with payload 38h (Test mode off), from LFAST Master

The peer LFAST device is required to maintain at least 2-bit IFG between two Loopback frames in this mode.

45.8.7.1.2 Rx LVDS LoopBack mode

This loopback mode is provided to verify and characterize the LVDS pads. In this loopback mode the data received by LFAST on Rx LVDS input is loopback to Tx LVDS output, bypassing LFAST. For LVDS loopback the output of the LR is passed to the LD via "Rx LVDS LoopBack Mux". Bit 0 of the header cannot be guaranteed to be asserted when the incoming data from the other device is looped back. In this loopback, the Rx Interface Controller operates in normal mode, decoding the frames (header, payloads) but the Tx Interface Controller is ignored.

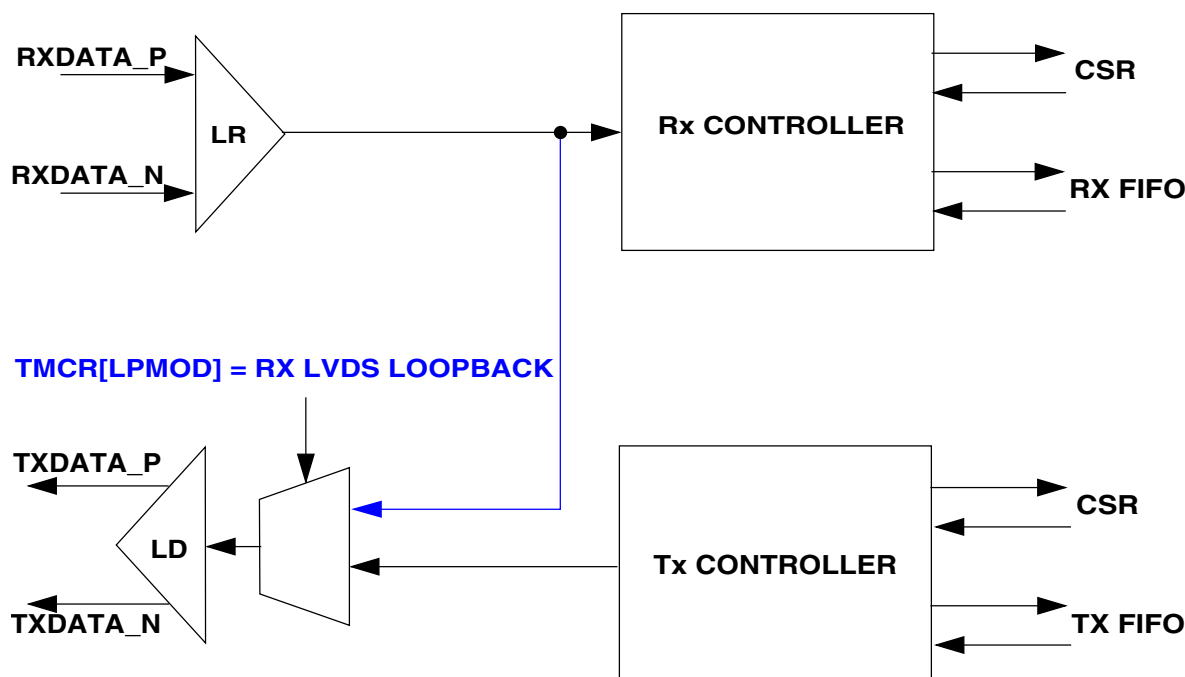


Figure 45-61. Rx LVDS LoopBack mode

Entry to Rx LVDS Loopback mode:

1. S/W programs $\text{TMCR}[\text{LPMOD}] = 001b$.
2. Loopback can be turned ON by either of the following methods:

- S/W writes $\text{TMCR}[\text{LPON}] = 1$.
- For Slave Only: - Reception of ICLC Frame with Payload = FFh (Loopback mode ON), from LFAST Master.

Exit from Rx LVDS Loopback mode can be done by any of the following methods:

- S/W programs $\text{TMCR}[\text{LPON}] = 0$.
- For LFAST Slave:
 - Transmission of ICLC frame with payload 38h (Test mode off), from LFAST Master.

45.8.7.1.3 Tx loopback mode without automatic frame generation

This loopback mode is provided to verify the LFAST functionality, if LVDS pads are not functional. In this loopback mode the data transmitted by LFAST on Tx LVDS output is loopbacked internally on Rx LVDS input, bypassing LVDS pads.

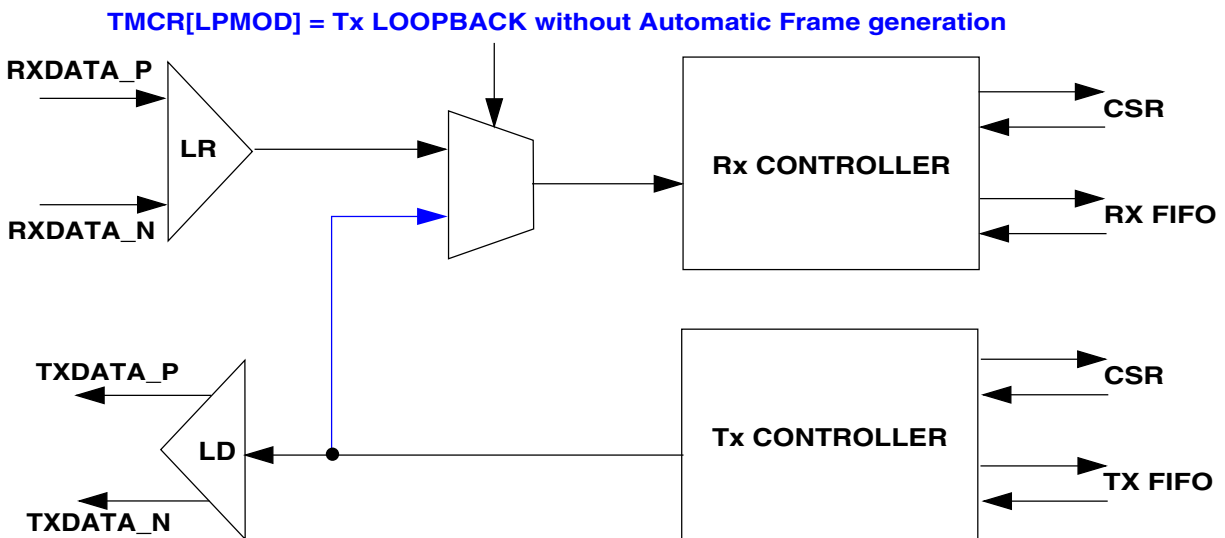


Figure 45-62. Tx LoopBack mode without automatic frame generation

Entry to Tx Loopback without Automatic frame generation mode:

1. S/W programs $\text{TMCR}[\text{LPMOD}] = 010b$.
2. Loopback can be turned on by either of the following methods:
 - S/W programs $\text{TMCR}[\text{LPON}] = 1$.
 - For Slave Only: Reception of ICLC frame with payload FFh (Loopback mode on), from LFAST Master

Exit from Tx Loopback without Automatic frame generation mode can be done by any of the following methods:

- S/W programs $TMCR[LPON] = 0$.
- For LFAST Slave: Reception of ICLC frame with payload 38h (Test mode off), from Tx interface (using unsolicited frame).

All frames and error flags are decoded in this mode.

45.8.7.1.4 Automatic frame generation

The LFAST module supports two Loopback modes where pre-defined frames are generated automatically by the Tx Interface and Rx Interface indicates its successful reception by dedicated signals and status registers. These modes are defined for BIST like check for the LFAST, with minimal S/W intervention.

The Control register used for these modes are:

- $ALCR[LPCNTEN]$ defines whether fixed number of auto loopback frames to be transmitted
- $ALCR[LPFMCNT]$ defines the number of loopback frames to be transmitted if $ALCR[LPCNTEN] = 1$.

The Status signals and register used for these modes are:

- $GSR[LPCSDV]$: Valid Synchronization received.
- $GSR[LPCHDV]$: Frame with Header of 13h received.
- $GSR[LPCPDV]$: Frame with Payload of CBh received.
- $GSR[LPTXDN]$: Number of Auto Loopback frame transmitted with valid Synchronization, Header (13h) and Payload (CBh) is equal to $ALCR[LPFMCNT]$.

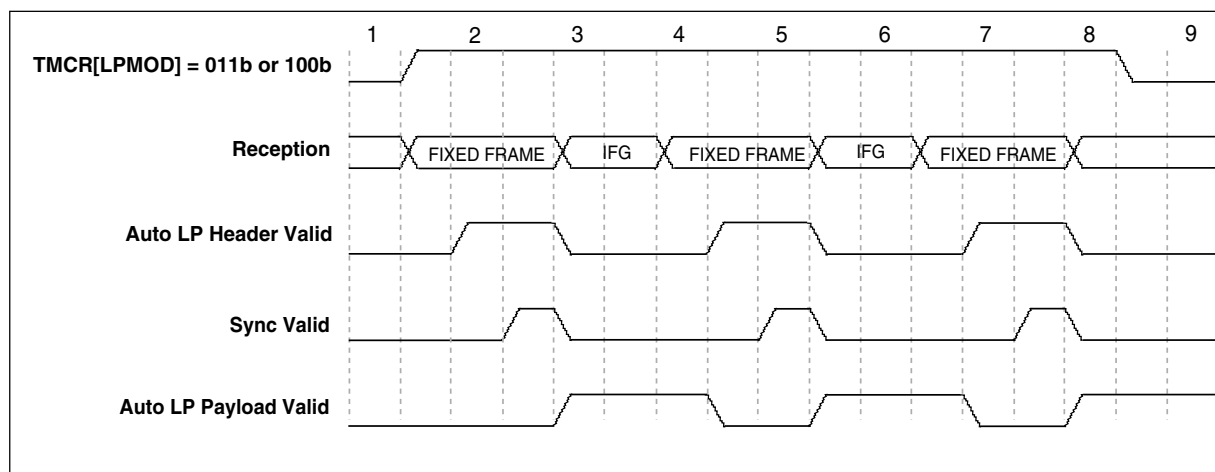


Figure 45-63. Automatic Loopback Test status signal timings

The two Loopback with automatic frame generation modes are:

1. Tx Loopback with Automatic frame generation
2. Tx LVDS (external) with Automatic frame generation

45.8.7.1.4.1 Tx loopback mode with automatic frame generation

When **TMCR[LPMOD] = 011b** (Tx Loopback mode with Automatic frame generation) the frames are generated by the Tx Interface. This mode helps to validate the Tx and Rx Path with minimal intervention from the S/W. The frames generated have fixed header of 13h and payload of CBh. The successful reception of this frame is indicated by the status signals and registers.

TMCR[LPMOD] = Tx LOOPBACK with Automatic Frame generation

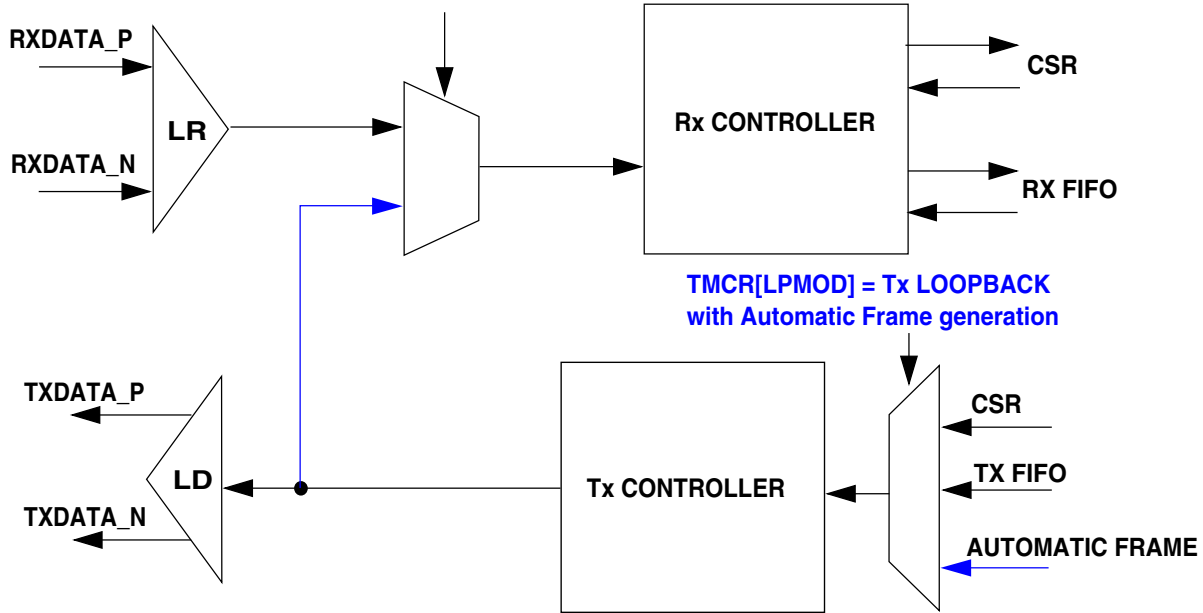


Figure 45-64. Tx LoopBack mode with automatic frame generation

Entry to Tx Loopback with automatic frame generation mode:

1. S/W programs TMCR[LPMOD] = 011b.
2. Loopback can be turned on by either of the following methods:
 - S/W programs TMCR[LPON] = 1.
 - For Slave Only: Reception of ICLC frame with payload FFh (Loopback mode on), from LFAST Master

Exit from Tx Loopback with Automatic frame generation mode can be done by any of the following methods:

- S/W programs TMCR[LPON] = 0.

45.8.7.1.4.2 Tx LVDS loopback (external) mode with automatic frame generation

In this mode the LD/Tx Controller is used to test the LR/Rx Controller. The idea is to use a test frame (predefined) from the Tx Controller out through the LD, loopback completed on the DUT-board (LD connected to LR via a transmission line), back into the LR, synchronized and correlated in the Rx Controller and compared to what was sent in the Downlink controller.

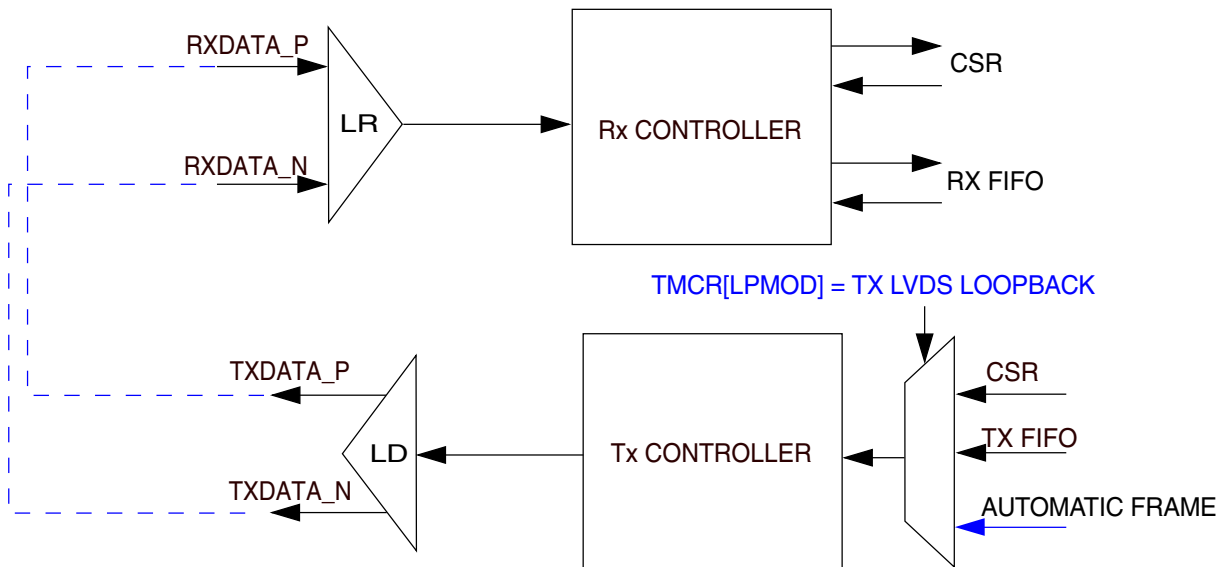


Figure 45-65. Tx LVDS loopback (external) mode with automatic frame generation

Entry to Tx LVDS loopback with automatic frame generation mode:

1. S/W programs $\text{TMCR}[\text{LPMOD}] = 100\text{b}$.
2. Loopback can be turned on by either of the following methods:
 - S/W programs $\text{TMCR}[\text{LPON}] = 1$.
 - For Slave Only: Reception of ICLC frame with payload FFh (Loopback mode on), from LFAST Master

Exit from Tx LVDS loopback with automatic frame generation mode can be done by the following method:

- S/W programs $\text{TMCR}[\text{LPON}] = 0$.

45.8.7.2 Clock test mode

The bit $\text{TMCR}[\text{CLKTST}]$ enables or disables the Clock Test mode of the LFAST module. In this mode the LFAST sends fixed pattern out on the LD at the current configured RxData clock rate. It is a RWM bitfield and the default setting is 0 (off). This bit can be set under one of the following conditions:

- S/W programs $\text{TMCR}[\text{CLKTST}]$.

The Tx Controller will send out a pattern of alternating 1 and 0 (pattern 101010...). This provides a divide by 2 test clock of the current Tx clock.

The Clock Test mode can be cancelled by either of the following methods:

- S/W programs $\text{TMCR}[\text{CLKTST}] = 0$.
- For LFAST Slave: Reception of ICLC frame with payload 38h (Test mode off) from LFAST Master

In clock test mode the Tx Controller does not output any synchronization pattern or header, just a pattern of alternating 1's and 0's.

This mode doesn't affect the functionality of the Rx Interface.

45.8.8 Interrupts

The LFAST module generates five interrupts to the processor:

- Transmit complete interrupt
- Transmit exception interrupt
- Receive complete interrupt
- Receive exception interrupt
- ICLC receive interrupt

Each interrupt is asserted when any one of their status conditions is met.

45.8.8.1 Transmit complete interrupt

This interrupt indicates a transfer completion of a frame. This interrupt is asserted whenever any of the following bits are set in the TISR and also the corresponding mask bits are set in the TIER.

- TXDTF
- TXICLCF
- TXUNSF
- TXPNGF

Each of these bits can be individually maskable. So, those bits that are not required to generate an interrupt to the processor can be masked. Once the interrupt is asserted, it can be negated by clearing the corresponding flag bit in the TISR.

45.8.8.2 Transmit exception interrupt

This interrupt indicates occurrence of an exception condition in the Tx block. This interrupt is asserted whenever any of the following bits are set in the TISR along with the corresponding mask bits in the TIER.

- TXOVF
- TXIEF

Each of these bits can be individually masked. So, those bits that are not required to generate an interrupt to the processor can be masked. Once the interrupt is asserted, it can be negated by clearing the corresponding flag bit in the TISR.

Table 45-55. Recommended transmit exception handling

Exception	Recommendation
TXOVF	• MCR[TXEN] = 0 • The system level interfaces transmit is reseted and enabled again • For Slave: After reception of ICLC Ping Response Request the MCR[TXEN] may be set • For Master: The MCR[TXEN] should be set and an ICLC frame Ping Response Request should be sent
TXIEF	• The MCR[TXEN] should be set

45.8.8.3 Receive complete interrupt

This interrupt indicates a reception of a frame. This interrupt is asserted whenever any of the following bits is set in the RISR along with the corresponding interrupt enable bit is also set in the RIER.

- RXCTSF
- RXDF
- RXUNSF

Each of these bits are individually maskable. So, the bits that are not required to generate an interrupt to the processor can be masked by clearing its bit in the RIER. Once the interrupt is asserted, it can be negated by clearing the corresponding flag bit in the RISR.

45.8.8.4 Receive exception interrupt

This interrupt indicates occurrence of an exception condition in Rx block. This interrupt is asserted whenever any of the following bits is set in the RISR and corresponding enable mask bit is also set in the RIER.

- RXLCEF
- RXICF
- RXSZF
- RXUOF
- RXUFF
- RXMXF
- RXMNF
- RXOFF

Each of these bits are individually maskable. So, those bits that are not required to generate an interrupt to the processor can be masked by clearing the appropriate bit in the RIER. For details on the cause of assertion of each bit refer to the RISR description. Once the interrupt is asserted, it can be negated by clearing the corresponding flag bit in the RISR.

Table 45-56. Recommended receive exception handling mechanism

Exception	Recommendation
RXLCEF	The MCR[RXEN] may be set and cleared, as this exception can result in loss of subsequent packet
RXICF	No action required
RXSZF	The MCR[RXEN] may be set and cleared, as this exception can result in loss of subsequent packet
RXOFF	• The MCR[RXEN] may be cleared • The system level interfaces receive may be reset and enabled, as current transfer may be corrupted
RXUFF	No action required
RXMXF	The system level interfaces receive should be enabled, if not enabled
RXMNF	No action required
RXUNSF	No action required

45.8.8.5 ICLC receive interrupt

This interrupt indicates reception of a valid ICLC frame. This interrupt is asserted whenever any of the following bits is set in the RIISR along with the corresponding enable bit mask bit is set in the RIIER.

- ICPONF
- ICPOFF
- ICTSF
- ICTFF
- ICRSF
- ICRFF
- ICTEF
- ICTDF
- ICCTF
- ICLPF
- ICTOF
- ICPRF
- ICPSF
- ICPFF

Each of these bits are individually maskable. So those bits that are not required to generate an interrupt to the processor can be masked. For details on the cause of assertion of each bit, refer RIISR. Once the interrupt is asserted, it can be negated by clearing the corresponding flag bit in RIISR.

45.9 Packet memory

The LFAST stores packet frames for transmission, and reads packet frames after reception. The transmitter has its own dedicated buffer and the receiver has its own dedicated buffer, and they are not shared between each other.

Table 45-57. Frames Supported by LFAST interfaces

Frame Type	Tx Buffer(in bits)	Rx Buffer(in bits)	Memory Type
Data Frame	38 × 32 max 6 packets	38 × 32 max 6 packets	FIFO
Unsolicited Frame	9 × 32 max 1 packet	9 × 32 max 1 packet	Registers UNSTD[8-0], UNSRDR[8-0]
ICLC Frame	1 × 7 max 1 packet ¹	1 × 8 max 1 packet ²	Registers ICR, PISR
CTS Frame	N/A	N/A	N/A

1. Only for MCR[MSEN] = 1

2. Only for ping response data when MCR[MSEN] = 1

45.10 Resets

The various blocks in LFAST are reset as described in the table below.

Table 45-58. Recommended receive exception handling mechanism

Reset	Blocks Reset
Asynchronous Hardware reset	Polarity: Active Low • Clock control module (CCM) • LFAST Domain Logic (Rx and Tx block) • System Side Module Interface FIFOs • LFAST Register Space
DRFRST bit of Mode Configuration Register (MCR)	Polarity: Active High • Clock control module (CCM) • LFAST Domain Logic (Rx and Tx block) • System Side Module Interface FIFOs • LFAST Register Space
DRFEN bit of Mode Configuration Register (MCR)	Polarity: Active Low • Clock control module (CCM) • LFAST Domain Logic (Rx and Tx block) • System Side Module Interface FIFOs • LFAST Status Registers • SCR[RDR] and SCR[TDR] • TMCR[CLKTST] and TMCR[LPON] • ICR[ICLCSEQ] and ICR[SNDCICLC] • PICR[PNGREQ] • UNSTCR[USNDRQ]

45.11 Clocks

The LFAST mainly works on three clocks:

- System Bus Clock used to program the registers.
- Protocol clock, which is either High Speed PLL clock or Low Speed clock depending on the speed mode, used for LFAST protocol operation.
- System Side Module clock, which is synchronous to the System Bus clock.

45.11.1 Clocking strategy

The following figure shows the clock domain in which each module functions. The PLL provides eight phases of the high speed clock to the Clock Muxing portion of the Clock Control Module. The Clock Module will then generate four phases of slow speed clock using both the edges of lfast_sysclk, muxes high speed and low speed clocks and provides a muxed clock to the Sampler Module.

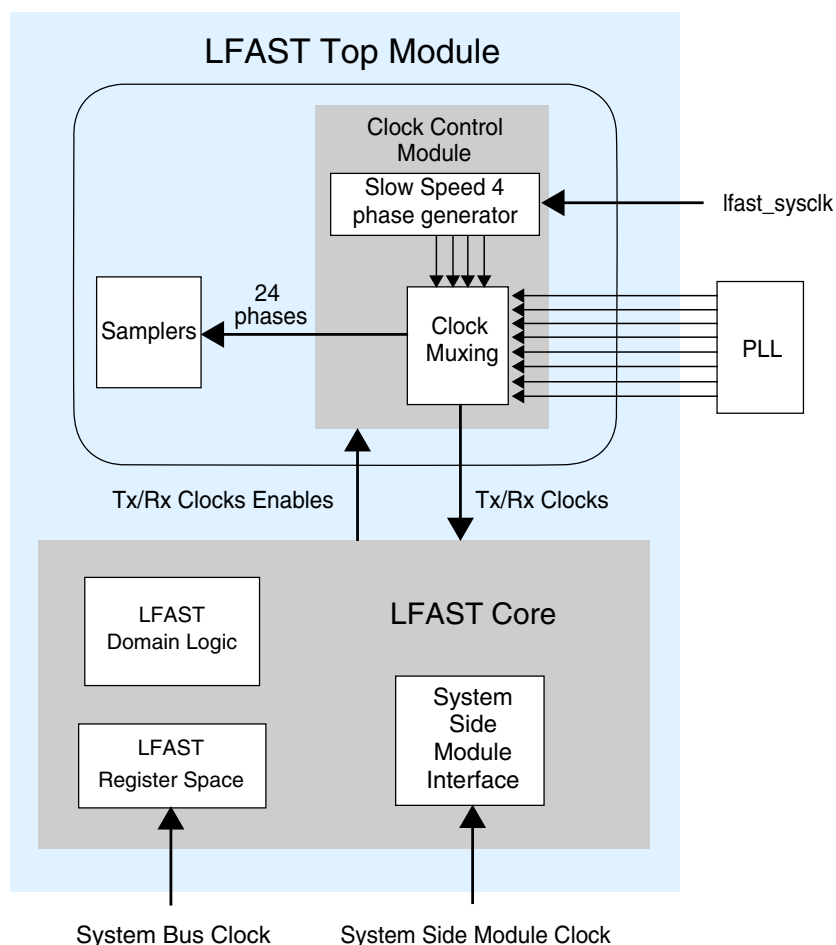


Figure 45-66. LFAST module clock domains

45.11.2 Slow speed clock

45.11.2.1 External muxing

The Clock Control Module will receive `lfast_sysclk` from the clocking subsystem as shown in [Figure 45-67](#), [Figure 45-68](#) and [Figure 45-69](#).

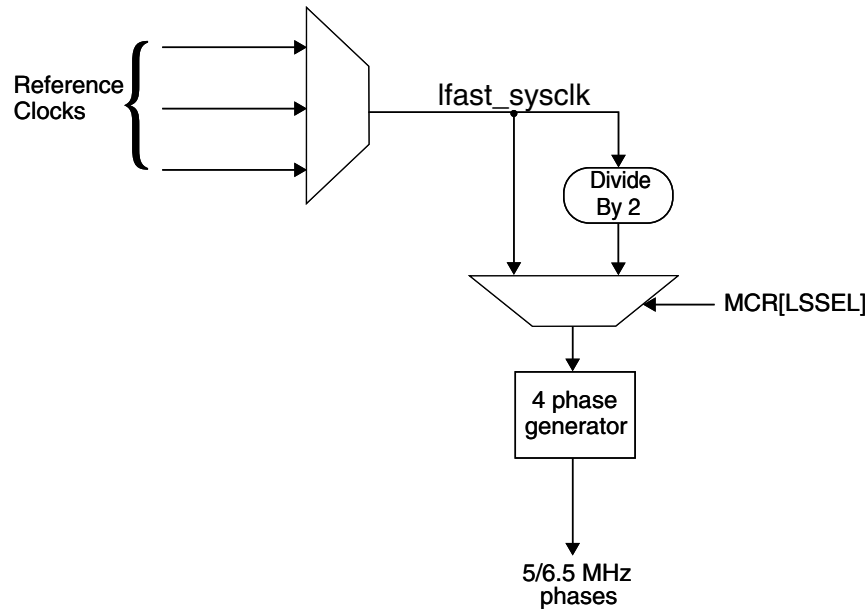


Figure 45-67. External clock muxing of lfast_sysclk

All the reference clocks will be muxed first to provide lfast_sysclk to the module and then either div/2 or direct muxed clock will be used to generate 4 slow speed phases in the Clock Control Module of LFAST as shown in [Figure 45-67](#).

The lfast_sysclk could be either 20/26 MHz or 10/13 MHz. When lfast_sysclk is 20/26 MHz frequency, the clock control module will generate 4 phases of slow speed clock of frequency lfast_sysclk/4 when MCR[LSSEL] = 1. If lfast_sysclk is 20/26 MHz it needs to be first divided by 2 before using it for 4 phase generation in LFAST Clock Module, which generates 4 phases of half of the input frequency as shown in [Figure 45-68](#) (selected clock path shown in green).

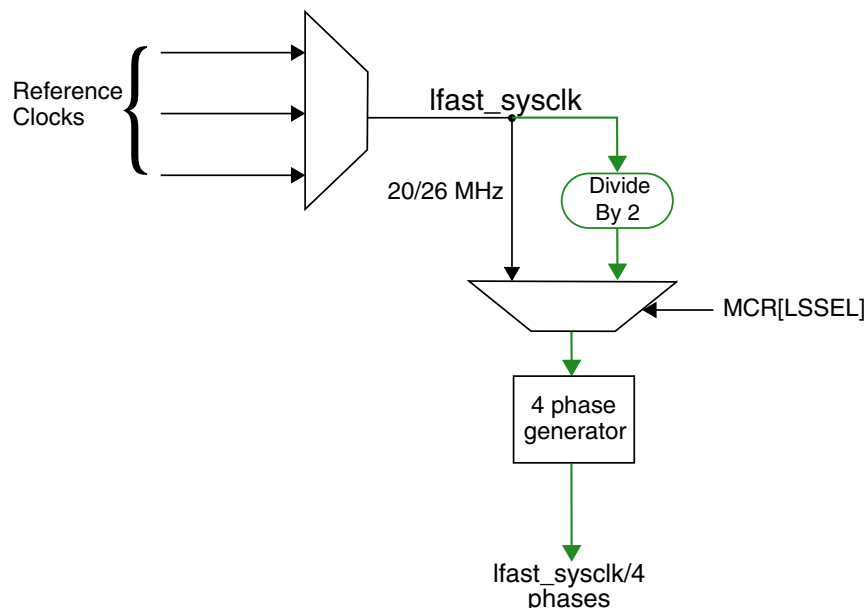


Figure 45-68. External clock muxing of lfast_sysclk of 20/26 MHz

In this case, lfast_sysclk is 10/13 MHz frequency, the clock control module will generate 4 phases of slow speed clock of frequency lfast_sysclk/2 when MCR[LSSEL] = 0. If lfast_sysclk is 10/13 MHz then it does not need to be first divided by 2 before using it for 4 phase generation in LFAST Clock Module, which generates 4 phases of half of the input frequency as shown in Figure 45-69 (selected clock path selected shown in magenta).

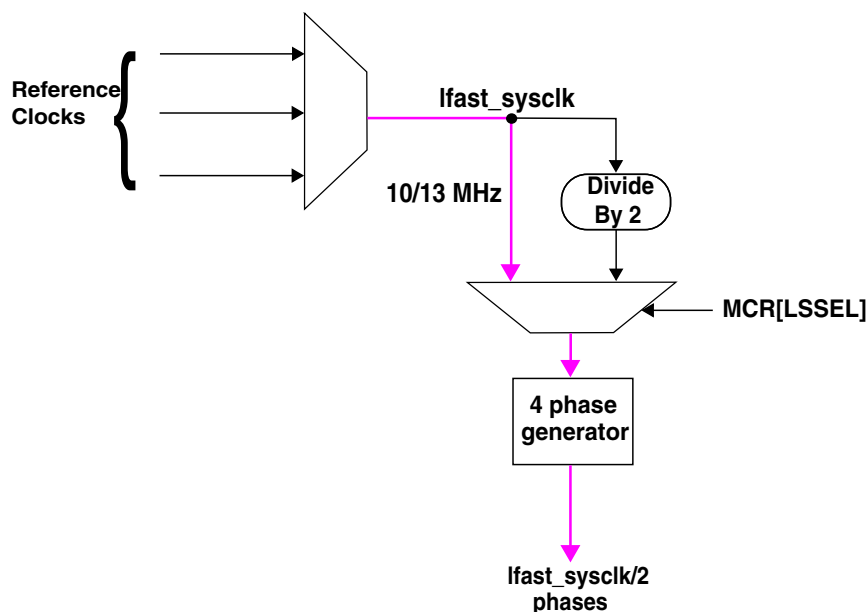


Figure 45-69. External clock muxing of lfast_sysclk of 10 MHz

45.11.2.2 Slow Speed 4 phase generator

Clock control module will generate 4 phases of slow speed clock of frequency of 10/13 MHz. For instance, if lfast_sysclk is 10 MHz then 4 phases of 5 MHz will be generated. The following figure shows 4 phases getting generated using lfast_sysclk.

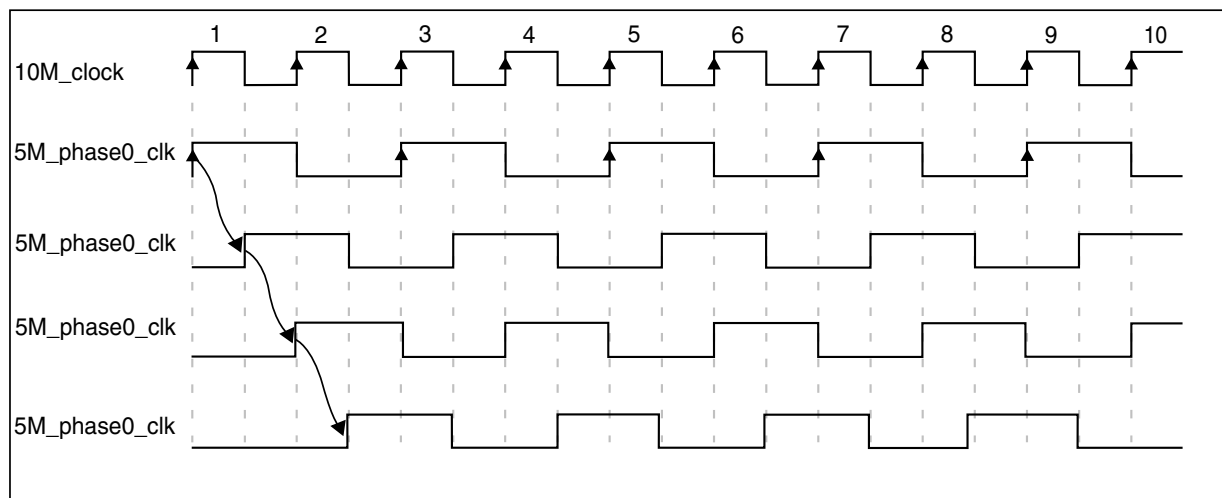


Figure 45-70. Slow speed 4 phases generation

45.11.3 Rx Controller Clocks

45.11.4 Clocking Module Requirements for High Speed Phases

The high speed phases are obtained from the PLL via the Clocking Module as shown in the following figure.

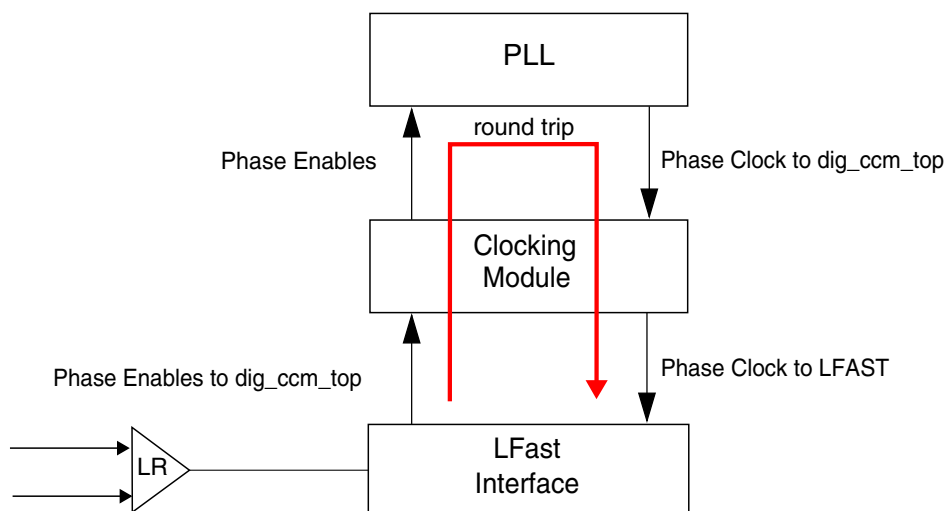


Figure 45-71. Clock enables and clock paths

The LFAST block will know which of the phases will be required for the different modes of operations. Therefore the LFAST block will provide enables and speed mode switches to the Clocking Module. The Clocking Module will use these signals to control the enables to the clock phases to reduce the power consumption.

The 8 phases of clocks signal are fed to the Clocking Module and muxed with the low speed phases. Depending on enables and data rate speed modes, the selected clocks are passed to the Sampler block and interface block.

COCR[PHSSEL] is connected to the Clocking Module, which selects whether 8 or 4 phases are selected for High Speed mode. It is only valid for high speed.

The routing of the 8 high speed phases to the interface is critical. Each clock phase will need to have the same routing track length from the PLL to the interface of Clocking Module. Each of the 8 high speed phases are separated by 45 degrees. This separation needs to be maintained from the phase generator to the interface.

45.11.5 Clock module requirements for low speed phases

45.11.5.1 Low speed

The low speed clock phases are generated in Clocking Module. These four phases are required to sample the data correctly at Low Speed mode. The LFAST block will provide the enable for the phases. The clock source for the low speed phases is SYSCLK. The Clocking Module block will need to generate the 4 phases of low speed clock 90 degrees apart.

Using the Low Speed mode on the interface allows the polyphase generation logic to be turned off and the requirement of high-speed-freq × 8 MHz clock from the PLL is not needed.

Table 45-59. Rx clocks summary

Rx Speed mode	Source
Low Speed	lfast_sysclk
High Speed	PLL

45.11.6 Tx Controller Clocks

45.11.6.1 Clocking Module Requirements for Speed Phases

The low speed phase 0 clock is sourced from the lfast_sysclk and the high speed phase 0 clock is sourced from the PLL. See the following table.

Table 45-60. Tx clocks summary

Tx Speed mode	Source
Low Speed	lfast_sysclk
High Speed	PLL

45.11.6.2 Transmit Clock Muxing

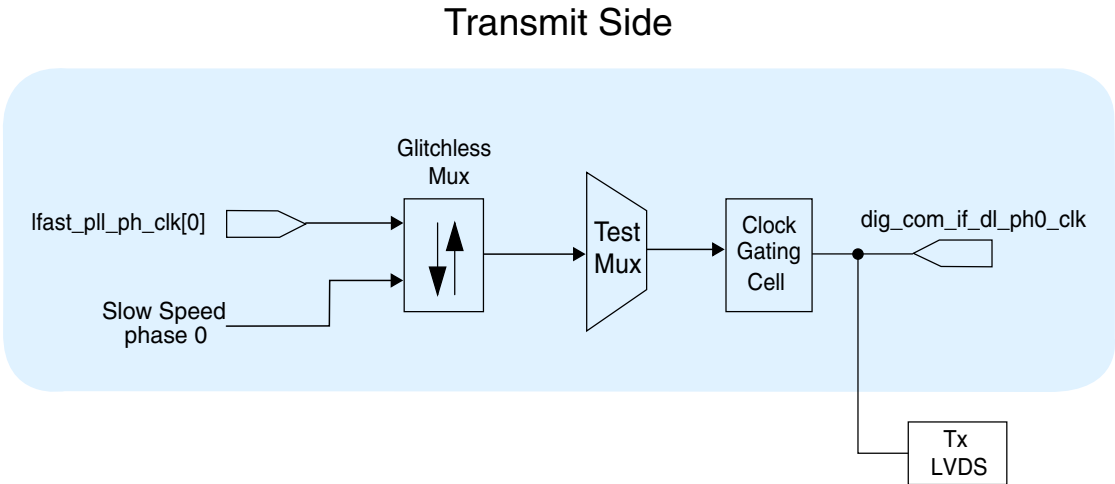


Figure 45-72. Transmit Clocks Muxing

45.11.6.3 Tx Request Clock Control

The Tx phase 0 clock is enabled when all the resets are negated.

Chapter 46

Fast Ethernet Controller (FEC)

46.1 Chip-specific FEC information

This chip has one instance of the FEC module.

46.1.1 FEC interface selection

On this chip, the FEC can operate using MII-Lite or RMII interfaces.

46.1.2 FEC clocking

See [Ethernet clocking](#) for information about FEC clocking restrictions on this chip.

46.2 Introduction

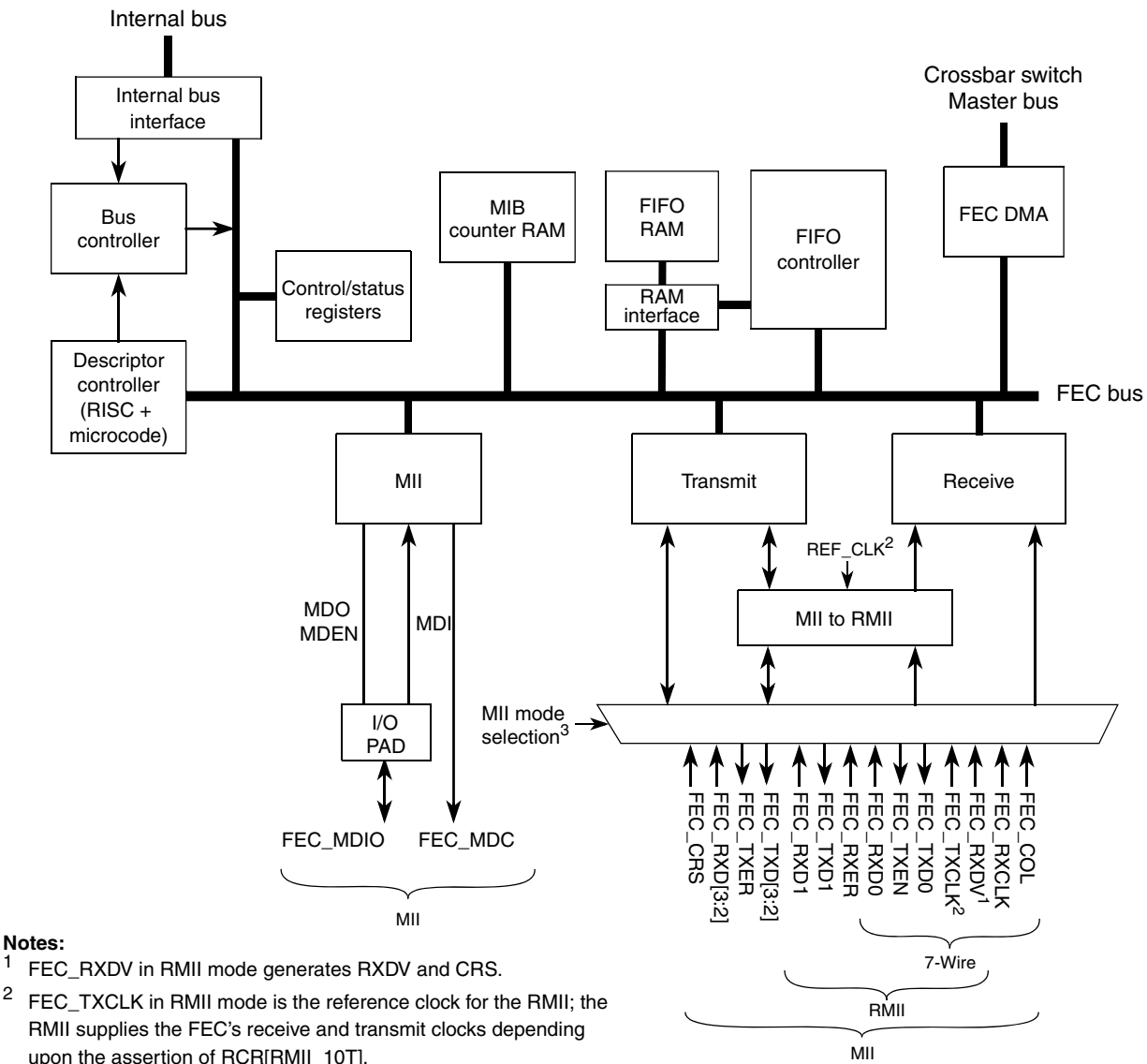
The Fast Ethernet Controller (FEC) is a communication controller that supports 10 and 100 Mbps Ethernet/IEEE 802.3 networks. An external transceiver interface and transceiver function are required to complete the interface to the media. The FEC supports five different standard MAC-PHY (physical) interfaces for connection to an external Ethernet transceiver. The FEC supports the 10/100 Mbps MII, 10/100 Mbps MII-Lite, 10/100 Mbps reduced MII, and the 10 Mbps-only 7-wire interface.

46.2.1 Block diagram

[Figure 46-1](#) shows the block diagram of the FEC. The FEC is implemented with a combination of hardware and microcode. The off-chip (Ethernet) interfaces are compliant with industry and IEEE 802.3 standards.

NOTE

When MII-Lite is used on this chip, the FEC_TXER, FEC_RXER, FEC_COL, and FEC_CRS signals shown in [Figure 46-1](#) are not used.



- Notes:**
- ¹ FEC_RXDV in RMII mode generates RXDV and CRS.
 - ² FEC_TXCLK in RMII mode is the reference clock for the RMII; the RMII supplies the FEC's receive and transmit clocks depending upon the assertion of RCR[RMII_10T].
 - ³ Interface selection is chip-specific; see the chapter that describes how modules are configured and connected.

Figure 46-1. FEC block diagram

The following table provides a brief description of the blocks in the previous figure.

Table 46-1. FEC sub-block descriptions

Block	Description
Control and status registers (CSR)	Provides global control (Ethernet reset and enable) and interrupt managing registers

Table continues on the next page...

Table 46-1. FEC sub-block descriptions (continued)

Block	Description
Descriptor controller	A RISC-based controller providing these functions in the FEC: • Initialization (those internal registers not initialized by you or hardware) • High-level control of the DMA channels (initiating DMA transfers) • Interpreting buffer descriptors • Address recognition for receive frames • Random number generation for transmit collision backoff timer
DMA block	Provides multiple channels allowing transmit data, transmit descriptor, receive data and receive descriptor accesses to all run independently Note: In this document, "DMA" refers to the DMA block within the FEC, and is not related to any other DMA controllers that may be present on the chip.
Media independent interface (MII)	Provides a serial channel for control/status communication with the external physical layer device (transceiver). This serial channel consists of: • The management data clock (FEC_MDC) • The management data input/output lines (FEC_MDIO) of the MII interface
Message information block (MIB)	Maintains counters for a variety of network events and statistics. It is not necessary for operation of the FEC, but provides valuable counters for network management. The counters supported are: • The RMON (RFC 1757) Ethernet Statistics group • Some of the IEEE 802.3 counters
RAM	Serves as the focal point of all data flow in the Fast Ethernet controller and divides into transmit and receive FIFOs. The FIFO boundaries are programmable using the FRSR register. User data flows to/from the DMA block from/to the receive/transmit FIFOs. Transmit data flows from the transmit FIFO into the transmit block. Receive data flows from the receive block into the receive FIFO.
Transmit and receive blocks	Provide the Ethernet MAC functionality (with some assist from microcode)

46.2.2 Features

The FEC incorporates the following features:

- Support for the following Ethernet physical interfaces:¹
 - 100-Mbps IEEE 802.3 MII
 - 10-Mbps IEEE 802.3 MII
 - 100-Mbps IEEE 802.3 MII-Lite

1. For interface selection options, see the chip-specific FEC information.

- 10-Mbps IEEE 802.3 MII-Lite
- 100-Mbps reduced media independent interface (RMII)
- 10-Mbps reduced media independent interface (RMII)
- 10-Mbps 7-wire interface (industry standard)
- IEEE 802.3 full duplex flow control
- Programmable max frame length supports IEEE 802.1 VLAN tags and priority
- Support for full-duplex operation (200 Mbps throughput) with a minimum internal bus clock rate of 50 MHz
- Support for half-duplex operation (100 Mbps throughput) with a minimum internal bus clock rate of 25 MHz
- Retransmission from transmit FIFO following a collision (no processor bus utilization)
- Automatic internal flushing of the receive FIFO for runts (collision fragments) and address recognition rejects (no processor bus utilization)
- Address recognition
 - Frames with broadcast address may be always accepted or always rejected
 - Exact match for single 48-bit individual (unicast) address
 - Hash (64-bit hash) check of individual (unicast) addresses
 - Hash (64-bit hash) check of group (multicast) addresses
 - Promiscuous mode

46.2.3 Clocking requirements

The FEC system clock frequency must be equal to or greater than the FEC protocol reference clock frequency (for example, 25 MHz system frequency for 100 Mbps protocol frequency). This requirement is true regardless of the reference selected by the application.

46.3 Modes of operation

This section describes the FEC operating modes.

46.3.1 Full and half duplex operation

Full duplex mode is for use on point-to-point links between switches or end node to switch. Half duplex mode works in connections between an end node and a repeater or between repeaters. TCR[FDEN] controls duplex mode selection.

When configured for full duplex mode, flow control may be enabled. For more details, see:

- The RFC_PAUSE and TFC_PAUSE fields in the TCR
- The RCR[FCE] field
- [Full duplex flow control](#)

46.3.2 Interface options

The following interface options are supported². A detailed discussion of the interface configurations is provided in [Network interface options](#).

46.3.2.1 10 Mbps and 100 Mbps MII interface

The IEEE 802.3 standard defines the media independent interface (MII) for 10/100 Mbps operation. The MAC-PHY interface may be configured to operate in MII mode by programming RCR[MII_MODE]=1.

The operation speed is determined by the FEC_TXCLK and FEC_RXCLK pins driven by the external transceiver. The transceiver auto-negotiates the speed or software controls it via the serial management interface (FEC_MDC/FEC_MDIO pins) to the transceiver. Refer to the MMFR and MSCR register descriptions, as well as the section on the MII, for a description of how to read and write registers in the transceiver via this interface.

46.3.2.2 10 Mbps and 100 Mbps MII-Lite interface

MII-Lite refers to MII without the following signals:

- FEC_COL
- FEC_CRs

2. For interface selection options, see the chip-specific FEC information.

Modes of operation

- FEC_RXER
- FEC_TXER

The MII-Lite connection is for PHYs that do not have all the MII pins and signals that are available for MII.

The following table provides additional information on the behavior of these signals when MII-Lite is used.

Table 46-2. MII signals not used in MII-Lite

Signal	Information
FEC_COL	This signal is asserted by the PHY when there has been a collision. With full-duplex links, this would always be low, so this signal's absence implies either that the link is full-duplex or that collisions are handled internally within the PHY. The signal is only asserted high on simultaneous Rx and Tx on half-duplex links. As this must be handled by the PHY if the signal is not available, it can be safely pulled low.
FEC CRS	Carrier sense is asserted high by the PHY to indicate that receive medium is non-idle. As in the case with the COL signal, this never occurs with full-duplex links. Again, it is recommended that this signal be pulled low using an external pulldown.
FEC_RXER	This signal is asserted high by the PHY to signal an error to the MAC. However, in these circumstances the PHY will deliberately corrupt the CRC of the incoming frame before passing to the MAC. This input to the FEC can therefore be pulled low and any RXER can be detected by a CRC error in the received frame where the CR bit is set in the receive buffer descriptor (RxBD).
FEC_TXER	This FEC output can be left unconfigured and unused. The signal was originally included for the test case in some PHY designs where the MAC layer in the FEC can demand that the PHY corrupt data during transmission. However, it is readily possible to do this within the MAC by setting the ABC (Append Bad CRC) bit in the transmit buffer descriptor (TxBD) for the frame being sent.

46.3.2.3 10 Mbps and 100 Mbps RMII interface

The reduced media independent interface (RMII) is a low-cost alternative to the IEEE802.3 MII standard. This interface provides the functionality of the MII interface on a total of 8 pins instead of 18. The RMII interface for 10/100 Ethernet MAC-PHY interface was defined by an industry consortium and is not currently included in the IEEE 802.3 standard. This functionality is selected as described in the chip configuration chapter, and is reflected in the read-only RCR[RMII_MODE] bit. The RCR[RMII_10T] bit determines the speed of operation. The reference clock for RMII is always 50 MHz, but this clock can be divided by 10 within the RCR register to support 10 Mbps operation. The PHY must be configured accordingly.

46.3.2.4 10 Mbps 7-wire interface operation

The FEC supports 7-wire interface used by many 10 Mbps Ethernet transceivers. The RCR[MII_MODE] bit controls this functionality. If this bit is cleared, MII mode is disabled and the 10 Mbps 7-wire mode is enabled.

46.3.3 Address recognition options

The address options supported are:

- Promiscuous
- Broadcast reject
- Individual address (hash or exact match)
- Multicast hash match

Address recognition options are discussed in detail in [Ethernet address recognition](#).

46.3.4 Internal loopback

Internal loopback mode is selected via RCR[LOOP]. Loopback mode is discussed in detail in [MII internal and external loopback](#).

46.4 External signal description

The following table describes the various FEC signals, as well as indicating which signals work in available modes.

Table 46-3. FEC signal descriptions

Signal name	MII	7-wire	RMII	Description
FEC_COL	X ¹	X	—	Asserted upon detection of a collision and remains asserted while the collision persists. This signal is not defined for full-duplex mode.
FEC_CRG	X ¹	—	—	Carrier sense. When asserted, indicates transmit or receive medium is not idle. In RMII mode, this signal is present on the FEC_RXDV pin.
FEC_MDC	X	—	X	Output clock provides a timing reference to the PHY for data transfers on the FEC_MDIO signal.
FEC_MDIO	X	—	X	Transfers control information between the external PHY and the media-access controller. Data is synchronous to FEC_MDC. This signal is an input after reset. When the FEC operates in 10Mbps 7-wire interface mode, this signal should be connected to VSS.

Table continues on the next page...

Table 46-3. FEC signal descriptions (continued)

Signal name	MII	7-wire	RMII	Description
FEC_RXCLK	X	X	—	Provides a timing reference for FEC_RXDV, FEC_RXD[3:0], and FEC_RXER.
FEC_RXDV	X	X	X	Asserting the FEC_RXDV input indicates PHY has valid nibbles present on the MII. FEC_RXDV must remain asserted from the first recovered nibble of the frame through to the last nibble. Assertion of FEC_RXDV must start no later than the SFD and exclude any EOF. In RMII mode, this pin also generates the CRS signal.
FEC_RXD0	X	X	X	This pin contains the Ethernet input data transferred from PHY to the media-access controller when FEC_RXDV is asserted.
FEC_RXD1	X	—	X	This pin contains the Ethernet input data transferred from PHY to the media access controller when FEC_RXDV is asserted.
FEC_RXD[3:2]	X	—	—	These pins contain the Ethernet input data transferred from PHY to the media access controller when FEC_RXDV is asserted.
FEC_RXER	X ¹	—	X	When asserted with FEC_RXDV, indicates PHY detects an error in the current frame. When FEC_RXDV is not asserted, FEC_RXER has no effect.
FEC_TXCLK	X	X	X	Input clock which provides a timing reference for FEC_TXEN, FEC_TXD[3:0] and FEC_TXER. In RMII mode, this signal is the reference clock for receive, transmit, and the control interface.
FEC_TXD0	X	X	X	The serial output Ethernet data and only valid during the assertion of FEC_TXEN.
FEC_TXD1	X	—	X	This pin contains the serial output Ethernet data and valid only during assertion of FEC_TXEN.
FEC_TXD[3:2]	X	—	—	These pins contain the serial output Ethernet data and valid only during assertion of FEC_TXEN.
FEC_TXEN	X	X	X	Indicates when valid nibbles are present on the MII. This signal is asserted with the first nibble of a preamble and is negated before the first FEC_TXCLK following the final nibble of the frame.
FEC_TXER	X ¹	—	—	When asserted for one or more clock cycles while FEC_TXEN is also asserted, PHY sends one or more illegal symbols. FEC_TXER has no effect at 10 Mbps or when FEC_TXEN is negated.
Note: Interface selection is chip-specific; see the chapter that describes how modules are configured and connected.				

1. This signal is not available in the MII-Lite interface.

46.5 Memory map and register definition

The FEC is programmed by a combination of control/status registers (CSRs) and buffer descriptors. The CSRs control operation modes and extract global status information. The descriptors pass data buffers and related buffer information between the hardware and software.

Each FEC implementation requires a 1 KB memory map space, which is divided into two sections of 512 bytes each for:

- Control/status registers
- Event/statistic counters held in the MIB block

The following table defines the top-level memory map.

Table 46-4. FEC memory map

Address offset	Function
000h–1FFh	Control/status registers
200h–3FFh	MIB block counters

The following table shows the FEC register map. Any address offset not explicitly mentioned in this table is reserved. Do not modify the content of reserved fields. All accesses to and from the registers must be via 32-bit accesses. There is no support for accesses other than 32-bit. Accessing reserved or unimplemented register spaces in the FEC will not return transfer errors.

The MIB counters (shown in the following table) define the locations in the MIB RAM space where hardware-maintained counters reside. The counters are divided into two groups:

- RMON counters include the Ethernet statistics counters defined in RFC 1757
- Truncated frames counter (only frames with lengths up to 2047 bytes are supported)

The transmit and receive RMON counters are independent, which ensures accurate network statistics when operating in full duplex mode.

The included IEEE counters support the mandatory and recommended counter packages defined in Section 5 of ANSI/IEEE Std. 802.3 (1998 edition). The FEC supports IEEE Basic Package objects, but these do not require counters in the MIB block. In addition, some of the recommended package objects supported do not require MIB counters. Counters for transmit and receive full duplex flow control frames are also included.

FEC memory map

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/page
4	Interrupt Event Register (FEC_EIR)	32	w1c	0000_0000h	46.5.1/2415
8	Interrupt Mask Register (FEC_EIMR)	32	R/W	0000_0000h	46.5.2/2417
10	Receive Descriptor Active Register (FEC_RDAR)	32	R/W	0000_0000h	46.5.3/2419
14	Transmit Descriptor Active Register (FEC_TDAR)	32	R/W	0000_0000h	46.5.4/2420
24	Ethernet Control Register (FEC_ECR)	32	R/W	F000_0000h	46.5.5/2421
40	MII Management Frame Register (FEC_MMFR)	32	R/W	Undefined	46.5.6/2422
44	MII Speed Control Register (FEC_MSCR)	32	R/W	0000_0000h	46.5.7/2423

Table continues on the next page...

FEC memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
64	MIB Control Register (FEC_MIBC)	32	R/W	C000_0000h	46.5.8/2424
84	Receive Control Register (FEC_RCR)	32	R/W	05EE_0001h	46.5.9/2426
C4	Transmit Control Register (FEC_TCR)	32	R/W	0000_0000h	46.5.10/2428
E4	Physical Address Low Register (FEC_PALR)	32	R/W	See section	46.5.11/2429
E8	Physical Address High Register and Type Field (FEC_PAUR)	32	R/W	See section	46.5.12/2430
EC	Opcode/Pause Duration (FEC_OPD)	32	R/W	See section	46.5.13/2430
118	Descriptor Individual Upper Address Register (FEC_IAUR)	32	R/W	See section	46.5.14/2431
11C	Descriptor Individual Lower Address Register (FEC_IALR)	32	R/W	See section	46.5.15/2431
120	Descriptor Group Upper Address Register (FEC_GAUR)	32	R/W	See section	46.5.16/2432
124	Descriptor Group Lower Address Register (FEC_GALR)	32	R/W	See section	46.5.17/2432
144	Transmit FIFO Watermark (FEC_TFWR)	32	R/W	0000_0000h	46.5.18/2433
14C	FIFO Receive Bound Register (FEC_FRBR)	32	R	0000_0600h	46.5.19/2434
150	FIFO Receive Start Register (FEC_FRSR)	32	R/W	0000_0500h	46.5.20/2434
180	Receive Descriptor Ring Start Register (FEC_ERDSR)	32	R/W	See section	46.5.21/2435
184	Transmit Buffer Descriptor Ring Start Register (FEC_ETDSR)	32	R/W	See section	46.5.22/2436
188	Receive Buffer Size Register (FEC_EMRBR)	32	R/W	See section	46.5.23/2436
200	Count of frames not counted correctly (FEC_RMON_T_DROP)	32	R/W	Undefined	46.5.24/2437
204	RMON Tx packet count (FEC_RMON_T_PACKETS)	32	R/W	Undefined	46.5.25/2438
208	RMON Tx broadcast packets (FEC_RMON_T_BC_PKT)	32	R/W	Undefined	46.5.26/2438
20C	RMON Tx multicast packets (FEC_RMON_T_MC_PKT)	32	R/W	Undefined	46.5.27/2439
210	RMON Tx packets with CRC/align error (FEC_RMON_T_CRC_ALIGN)	32	R/W	Undefined	46.5.28/2439
214	RMON Tx packets < 64 bytes, good CRC (FEC_RMON_T_UNDERSIZE)	32	R/W	Undefined	46.5.29/2440
218	RMON Tx packets > MAX_FL bytes, good CRC (FEC_RMON_T_OVERSIZE)	32	R/W	Undefined	46.5.30/2440

Table continues on the next page...

FEC memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
21C	RMON Tx packets < 64 bytes, bad CRC (FEC_RMON_T_FRAG)	32	R/W	Undefined	46.5.31/2441
220	RMON Tx packets > MAX_FL bytes, bad CRC (FEC_RMON_T_JAB)	32	R/W	Undefined	46.5.32/2441
224	RMON Tx collision count (FEC_RMON_T_COL)	32	R/W	Undefined	46.5.33/2442
228	RMON Tx 64 byte packets (FEC_RMON_T_P64)	32	R/W	Undefined	46.5.34/2442
22C	RMON Tx 65 to 127 byte packets (FEC_RMON_T_P65TO127)	32	R/W	Undefined	46.5.35/2443
230	RMON Tx 128 to 255 byte packets (FEC_RMON_T_P128TO255)	32	R/W	Undefined	46.5.36/2443
234	RMON Tx 256 to 511 byte packets (FEC_RMON_T_P256TO511)	32	R/W	Undefined	46.5.37/2444
238	RMON Tx 512 to 1023 byte packets (FEC_RMON_T_P512TO1023)	32	R/W	Undefined	46.5.38/2444
23C	RMON Tx 1024 to 2047 byte packets (FEC_RMON_T_P1024TO2047)	32	R/W	Undefined	46.5.39/2445
240	RMON Tx packets with > 2048 bytes (FEC_RMON_T_P_GTE2048)	32	R/W	Undefined	46.5.40/2445
244	RMON Tx Octets (FEC_RMON_T_OCTETS)	32	R/W	Undefined	46.5.41/2446
248	Count of transmitted frames not counted correctly (FEC_IEEE_T_DROP)	32	R/W	Undefined	46.5.42/2446
24C	Frames transmitted OK (FEC_IEEE_T_FRAME_OK)	32	R/W	Undefined	46.5.43/2447
250	Frames transmitted with single collision (FEC_IEEE_T_1COL)	32	R/W	Undefined	46.5.44/2447
254	Frames transmitted with multiple collisions (FEC_IEEE_T_MCOL)	32	R/W	Undefined	46.5.45/2448
258	Frames transmitted after deferral delay (FEC_IEEE_T_DEF)	32	R/W	Undefined	46.5.46/2448
25C	Frames transmitted with late collision (FEC_IEEE_T_LCOL)	32	R/W	Undefined	46.5.47/2449
260	Frames transmitted with excessive collisions (FEC_IEEE_T_EXCOL)	32	R/W	Undefined	46.5.48/2449
264	Frames transmitted with Tx FIFO underrun (FEC_IEEE_T_MACERR)	32	R/W	Undefined	46.5.49/2450
268	Frames transmitted with carrier sense error (FEC_IEEE_T_CSERR)	32	R/W	Undefined	46.5.50/2450
26C	Frames transmitted with SQE error (FEC_IEEE_T_SQE)	32	R/W	Undefined	46.5.51/2451
270	Flow control pause frames transmitted (FEC_IEEE_T_FDXFC)	32	R/W	Undefined	46.5.52/2451

Table continues on the next page...

FEC memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
274	Octet count for frames transmitted without error (FEC_IEEE_T_OCTETS_OK)	32	R/W	Undefined	46.5.53/2452
280	Count of received frames not counted correctly (FEC_RMON_R_DROP)	32	R/W	Undefined	46.5.54/2452
284	RMON Rx packet count (FEC_RMON_R_PACKETS)	32	R/W	Undefined	46.5.55/2453
288	RMON Rx broadcast packets (FEC_RMON_R_BC_PKT)	32	R/W	Undefined	46.5.56/2453
28C	RMON Rx multicast packets (FEC_RMON_R_MC_PKT)	32	R/W	Undefined	46.5.57/2454
290	RMON Rx packets with CRC/Align error (FEC_RMON_R_CRC_ALIGN)	32	R/W	Undefined	46.5.58/2454
294	RMON Rx packets < 64 bytes, good CRC (FEC_RMON_R_UNDERSIZE)	32	R/W	Undefined	46.5.59/2455
298	RMON Rx packets > MAX_FL bytes, good CRC (FEC_RMON_R_OVERSIZE)	32	R/W	Undefined	46.5.60/2455
29C	RMON Rx packets < 64 bytes, bad CRC (FEC_RMON_R_FRAG)	32	R/W	Undefined	46.5.61/2456
2A0	RMON Rx packets > MAX_FL bytes, bad CRC (FEC_RMON_R_JAB)	32	R/W	Undefined	46.5.62/2456
2A4	Reserved (FEC_RMON_R_RESVD_0)	32	R/W	Undefined	46.5.63/2457
2A8	RMON Rx 64 byte packets (FEC_RMON_R_P64)	32	R/W	Undefined	46.5.64/2457
2AC	RMON Rx 65 to 127 byte packets (FEC_RMON_R_P65TO127)	32	R/W	Undefined	46.5.65/2458
2B0	RMON Rx 128 to 255 byte packets (FEC_RMON_R_P128TO255)	32	R/W	Undefined	46.5.66/2458
2B4	RMON Rx 256 to 511 byte packets (FEC_RMON_R_P256TO511)	32	R/W	Undefined	46.5.67/2459
2B8	RMON Rx 512 to 1023 byte packets (FEC_RMON_R_P512TO1023)	32	R/W	Undefined	46.5.68/2459
2BC	RMON Rx 1024 to 2047 byte packets (FEC_RMON_R_P1024TO2047)	32	R/W	Undefined	46.5.69/2460
2C0	RMON Rx packets with > 2048 bytes (FEC_RMON_R_P_GTE2048)	32	R/W	Undefined	46.5.70/2460
2C4	RMON Rx octets (FEC_RMON_R_OCTETS)	32	R/W	Undefined	46.5.71/2461
2C8	Count of received frames not counted correctly (FEC_IEEE_R_DROP)	32	R/W	Undefined	46.5.72/2461
2CC	Frames received OK (FEC_IEEE_R_FRAME_OK)	32	R/W	Undefined	46.5.73/2462
2D0	Frames received with CRC error (FEC_IEEE_R_CRC)	32	R/W	Undefined	46.5.74/2462

Table continues on the next page...

FEC memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/page
2D4	Frames received with alignment error (FEC_IEEE_R_ALIGN)	32	R/W	Undefined	46.5.75/2463
2D8	Receive FIFO overflow count (FEC_IEEE_R_MACERR)	32	R/W	Undefined	46.5.76/2463
2DC	Flow control pause frames received (FEC_IEEE_R_FDXFC)	32	R/W	Undefined	46.5.77/2464
2E0	Octet count for frames received without error (FEC_IEEE_R_OCTETS_OK)	32	R/W	Undefined	46.5.78/2464

46.5.1 Interrupt Event Register (FEC_EIR)

When an event occurs that sets a bit in EIR, an interrupt occurs if the corresponding bit in the interrupt mask register (EIMR) is also set. Writing a 1 to an EIR bit clears it; writing 0 has no effect. This register is cleared upon hardware reset.

These interrupts can be divided into operational interrupts, transceiver/network error interrupts, and internal error interrupts. Interrupts which may occur in normal operation are GRA, TXF, TXB, RXF, RXB, and MII. Interrupts resulting from errors/problems detected in the network or transceiver are HBERR, BABR, BABT, LC, and RL. Interrupts resulting from internal errors are HBERR and UN.

Some of the error interrupts are independently counted in the MIB block counters:

- HBERR - IEEE_T_SQE
- BABR - RMON_R_OVERSIZE (good CRC), RMON_R_JAB (bad CRC)
- BABT - RMON_T_OVERSIZE (good CRC), RMON_T_JAB (bad CRC)
- LATE_COL - IEEE_T_LCOL
- COL_RETRY_LIM - IEEE_T_EXCOL
- XFIFO_UN - IEEE_T_MACERR

Software may choose to mask off these interrupts because these errors are visible to network management via the MIB counters.

Memory map and register definition

Address: 0h base + 4h offset = 4h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	HBERR	BABR	BABT	GRA	TXF	TXB	RXF	RXB	MII	EBERR	LC	RL	UN	0		
W	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c			
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

FEC_EIR field descriptions

Field	Description
0 HBERR	Heartbeat error. Indicates TCR[HBC] is set and that the COL input was not asserted within the heartbeat window following a transmission.
1 BABR	Babbling receive error. Indicates a frame was received with length in excess of RCR[MAX_FL] bytes.
2 BABT	Babbling transmit error. Indicates the transmitted frame length exceeds RCR[MAX_FL] bytes. Usually this condition is caused by a frame that is too long is placed into the transmit data buffer(s). Truncation does not occur.
3 GRA	Graceful stop complete. Indicates the graceful stop is complete. During graceful stop the transmitter is placed into a pause state after completion of the frame currently being transmitted. This bit is set by one of these conditions: • A graceful stop initiated by the setting of the TCR[GTS] bit is now complete. • A graceful stop initiated by the setting of the TCR[TFC_PAUSE] bit is now complete. • A graceful stop initiated by the reception of a valid full duplex flow control pause frame is now complete. See Full duplex flow control.
4 TXF	Transmit frame interrupt. Indicates a frame has been transmitted and the last corresponding buffer descriptor has been updated.
5 TXB	Transmit buffer interrupt. Indicates a transmit buffer descriptor has been updated.
6 RXF	Receive frame interrupt. Indicates a frame has been received and the last corresponding buffer descriptor has been updated.
7 RXB	Receive buffer interrupt. Indicates a receive buffer descriptor not the last in the frame has been updated.
8 MII	MII interrupt. Indicates the MII has completed the data transfer requested.
9 EBERR	Ethernet bus error. Indicates a system bus error occurred when a DMA transaction is underway. When the EBERR bit is set, ECR[ETHER_EN] is cleared, halting frame processing by the FEC. When this occurs, software needs to ensure that the FIFO controller and DMA also soft reset.

Table continues on the next page...

FEC_EIR field descriptions (continued)

Field	Description
10 LC	Late collision. Indicates a collision occurred beyond the collision window (slot time) in half duplex mode. The frame truncates with a bad CRC and the remainder of the frame is discarded.
11 RL	Collision retry limit. Indicates a collision occurred on each of 16 successive attempts to transmit the frame. The frame is discarded without being transmitted and transmission of the next frame commences. This error can only occur in half duplex mode.
12 UN	Transmit FIFO underrun. Indicates the transmit FIFO became empty before the complete frame was transmitted. A bad CRC is appended to the frame fragment and the remainder of the frame is discarded.
13–31 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

46.5.2 Interrupt Mask Register (FEC_EIMR)

The EIMR controls which interrupt events are allowed to generate actual interrupts. A hardware reset clears this register. If the corresponding bits in the EIR and EIMR registers are set, an interrupt is generated. The interrupt signal remains asserted until a 1 is written to the EIR bit (write 1 to clear) or a 0 is written to the EIMR bit.

Each bit in this register corresponds to an interrupt source defined by the EIR. The corresponding EIMR bit determines whether an interrupt condition can generate an interrupt. At every processor clock, the EIR samples the signal generated by the interrupting source. The corresponding EIR bit reflects the state of the interrupt signal even if the corresponding EIMR bit is set.

Address: 0h base + 8h offset = 8h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R																
W	HBERR	BABR	BABT	GRA	TXF	TXB	RXF	RXB	MII	EBERR	LC	RL	UN			
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R																
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

FEC_EIMR field descriptions

Field	Description
0 HBERR	Heartbeat error interrupt mask.

Table continues on the next page...

FEC_EIMR field descriptions (continued)

Field	Description
	0 The interrupt source is masked. 1 The interrupt source is not masked.
1 BABR	Babbling receive error interrupt mask. 0 The interrupt source is masked. 1 The interrupt source is not masked.
2 BABT	Babbling transmit error interrupt mask. 0 The interrupt source is masked. 1 The interrupt source is not masked.
3 GRA	Graceful stop complete interrupt mask. 0 The interrupt source is masked. 1 The interrupt source is not masked.
4 TXF	Transmit frame interrupt mask. 0 The interrupt source is masked. 1 The interrupt source is not masked.
5 TXB	Transmit buffer interrupt mask. 0 The interrupt source is masked. 1 The interrupt source is not masked.
6 RXF	Receive frame interrupt mask. 0 The interrupt source is masked. 1 The interrupt source is not masked.
7 RXB	Receive buffer interrupt mask. 0 The interrupt source is masked. 1 The interrupt source is not masked.
8 MII	MII interrupt mask. 0 The interrupt source is masked. 1 The interrupt source is not masked.
9 EBERR	Ethernet bus error interrupt mask. 0 The interrupt source is masked. 1 The interrupt source is not masked.
10 LC	Late collision interrupt mask. 0 The interrupt source is masked. 1 The interrupt source is not masked.
11 RL	Collision retry limit interrupt mask. 0 The interrupt source is masked. 1 The interrupt source is not masked.
12 UN	Transmit FIFO underrun interrupt mask.

Table continues on the next page...

FEC_EIMR field descriptions (continued)

Field	Description
	0 The interrupt source is masked. 1 The interrupt source is not masked.
13–31 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

46.5.3 Receive Descriptor Active Register (FEC_RDAR)

RDAR is a command register, written by the user, indicating the receive descriptor ring is updated (the driver produced empty receive buffers with the empty bit set).

When the register is written, the RDAR bit is set. This is independent of the data actually written by the user. When set, the FEC polls the receive descriptor ring and processes receive frames (if ECR[ETHER_EN] is also set). After the FEC polls a receive descriptor whose empty bit is not set, FEC clears the RDAR bit and ceases receive descriptor ring polling until the user sets the bit again, signifying that additional descriptors are placed into the receive descriptor ring.

The RDAR is cleared at reset and when ECR[ETHER_EN] is cleared.

Address: 0h base + 10h offset = 10h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0							RDAR	0							
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

FEC_RDAR field descriptions

Field	Description
0–6 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
7 RDAR	Set to 1 when this register is written, regardless of the value written. Cleared by the FEC when no additional empty descriptors remain in the receive ring. Also cleared when ECR[ETHER_EN] is cleared.
8–31 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

46.5.4 Transmit Descriptor Active Register (FEC_TDAR)

The TDAR is a command register which the user writes to indicate the transmit descriptor ring is updated (transmit buffers have been produced by the driver with the ready bit set in the buffer descriptor).

When the register is written, the TDAR field is set. This value is independent of the data actually written by the user. When set, the FEC polls the transmit descriptor ring and processes transmit frames (provided ECR[ETHER_EN] is also set). After the FEC polls a transmit descriptor that is a ready bit not set, FEC clears the TDAR field and ceases transmit descriptor ring polling until the user sets the bit again, signifying additional descriptors are placed into the transmit descriptor ring.

The TDAR field is cleared when any of the following events occur:

- Chip reset
- When ECR[ETHER_EN] is cleared
- When ECR[RESET] is set

Address: 0h base + 14h offset = 14h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0							TDAR	0							
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

FEC_TDAR field descriptions

Field	Description
0–6 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
7 TDAR	Set to 1 when this register is written, regardless of the value written. Cleared by the FEC device when no additional ready descriptors remain in the transmit ring. Also cleared when ECR[ETHER_EN] is cleared.
8–31 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

46.5.5 Ethernet Control Register (FEC_ECR)

ECR is a read/write user register, though hardware may alter fields in this register as well. The ECR enables/disables the FEC.

Address: 0h base + 24h offset = 24h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0				0	0	0									
W																
Reset	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0													0	ETHER_EN	RESET
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

FEC_ECR field descriptions

Field	Description
0–3 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
4 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
5 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
6–28 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
29 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
30 ETHER_EN	When this bit is set, FEC is enabled, and reception and transmission are possible. When this bit is cleared, reception immediately stops and transmission stops after a bad CRC is appended to any currently transmitted frame. The buffer descriptor(s) for an aborted transmit frame are not updated after clearing this bit. When ETHER_EN is cleared, the DMA, buffer descriptor, and FIFO control logic are reset, including the buffer descriptor and FIFO pointers. Hardware alters the ETHER_EN bit under the following conditions: • ECR[RESET] is set by software, in which case ETHER_EN is cleared. • An error condition causes the EIR[EBERR] bit to set, in which case ETHER_EN is cleared.
31 RESET	When this bit is set, the equivalent of a hardware reset is performed but it is local to the FEC. ECR[ETHER_EN] is cleared and all other FEC registers take their reset values. Also, any transmission/reception currently in progress is abruptly aborted. This bit is automatically cleared by hardware during the reset sequence. The reset sequence takes approximately eight internal bus clock cycles after this bit is set.

46.5.6 MII Management Frame Register (FEC_MMFR)

The MMFR is used to communicate with the attached MII compatible PHY device(s), providing read/write access to their MII registers. Performing a write to the MMFR causes a management frame to be sourced unless the MSCR is programmed to 0. If MSCR is cleared while MMFR is written and then MSCR is written with a non-zero value, an MII frame is generated with the data previously written to the MMFR. This allows MMFR and MSCR to be programmed in either order if MSCR is currently zero.

The MMFR does not reset to any predefined value.

To perform a read or write operation on the MII Management Interface, write the MMFR. To generate a valid read or write management frame, ST field must be written with a 01 pattern, and the TA field must be written with a 10. If other patterns are written to these fields, a frame is generated, but does not comply with the IEEE 802.3 MII definition.

To generate an IEEE 802.3-compliant MII Management Interface write frame (write to a PHY register), write {01 01 PHYAD REGAD 10 DATA} to the MMFR. Writing this pattern causes the control logic to shift out the data in the MMFR following a preamble generated by the control state machine. During this time, contents of the MMFR register are altered as the contents are serially shifted and are unpredictable if read by the user. After the write management frame operation completes, the MII interrupt is generated. At this time, contents of the MMFR match the original value written.

To generate an MII management interface read frame (read a PHY register), write {01 10 PHYAD REGAD 10 XXXX} to the MMFR (the content of the DATA field is unimportant). Writing this pattern causes the control logic to shift out the data in the MMFR following a preamble generated by the control state machine. During this time, contents of the MMFR register are altered as the contents are serially shifted and are unpredictable if read by the user. After the read management frame operation completes, the MII interrupt is generated. At this time, the contents of the MMFR register match the original value written except for the DATA field whose contents are replaced by the value read from the PHY register.

If the MMFR register is written while frame generation is in progress, the frame contents are altered. Software must use the MII interrupt to avoid writing to the MMFR while frame generation is in progress.

Address: 0h base + 40h offset = 40h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	ST		OP		PA				RA						TA		DATA															
W	ST		OP		PA				RA						TA		DATA															
Reset	X*	X*	X*	X*	X*	X*	X*	X*	X*	X*	X*	X*	X*	X*	X*	X*	X*	X*	X*	X*	X*	X*	X*	X*	X*	X*	X*	X*	X*	X*	X*	

* Notes:

- x = Undefined at reset.

FEC_MMFR field descriptions

Field	Description
0–1 ST	Start of frame delimiter. These bits must be programmed to 0b01 for a valid MII management frame.
2–3 OP	Operation code. 00 Write frame operation, but not MII compliant. 01 Write frame operation for a valid MII management frame. 10 Read frame operation for a valid MII management frame. 11 Read frame operation, but not MII compliant.
4–8 PA	PHY address. This field specifies one of up to 32 attached PHY devices.
9–13 RA	Register address. This field specifies one of up to 32 registers within the specified PHY device.
14–15 TA	Turn around. This field must be programmed to 10 to generate a valid MII management frame.
16–31 DATA	Management frame data. This is the field for data to be written to or read from the PHY register.

46.5.7 MII Speed Control Register (FEC_MSCR)

The MSCR:

- Provides control of the MII clock (FEC_MDC pin) frequency
- Allows a preamble drop on the MII management frame

The MII_SPEED field must be programmed with a value to provide an FEC_MDC frequency of less than or equal to 2.5 MHz to be compliant with the IEEE 802.3 MII specification. The MII_SPEED must be set to a non-zero value to source a read or write management frame. After the management frame is complete, the MSCR register may optionally be set to 0 to turn off the FEC_MDC. The FEC_MDC generated has a 50% duty cycle except when MII_SPEED changes during operation (change takes effect following a rising or falling edge of FEC_MDC).

If the internal bus clock is 25 MHz, programming this register to 0000_0005h results in an FEC_MDC as stated the equation below.

$$25 \text{ MHz} \times \frac{1}{52} = 2.5 \text{ MHz}$$

Figure 46-9. FEC_MDC example

[Table 46-13](#) shows optimum values for MII_SPEED as a function of internal bus clock frequency.

Table 46-13. Programming examples for MSCR

Internal FEC clock frequency	MSCR[MII_SPEED]	FEC_MDC frequency
25 MHz	5h	2.50 MHz
33 MHz	7h	2.36 MHz
40 MHz	8h	2.50 MHz
50 MHz	Ah	2.50 MHz
66 MHz	Eh	2.36 MHz
80 MHz	10h	2.50 MHz
83 MHz	10h	2.59 MHz

Address: 0h base + 44h offset = 44h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15				
R	0																			
W																				
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0				
Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31				
R	0								DIS_ PRE	MII_SPEED										Reserved
W																				
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0				

FEC_MSCR field descriptions

Field	Description
0–23 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
24 DIS_PRE	Setting this bit causes the preamble (32 ones) not to be prepended to the MII management frame. The MII standard allows the preamble to be dropped if the attached PHY device(s) does not require it.
25–30 MII_SPEED	Controls the frequency of the MII management interface clock (FEC_MDC) relative to the internal bus clock. A value of 0 in this field turns off the FEC_MDC and leaves it in low voltage state. Any non-zero value results in the FEC_MDC frequency of $1/(\text{MII\_SPEED} \times 2)$ of the internal bus frequency.
31 Reserved	This field is reserved. Reserved

46.5.8 MIB Control Register (FEC_MIBC)

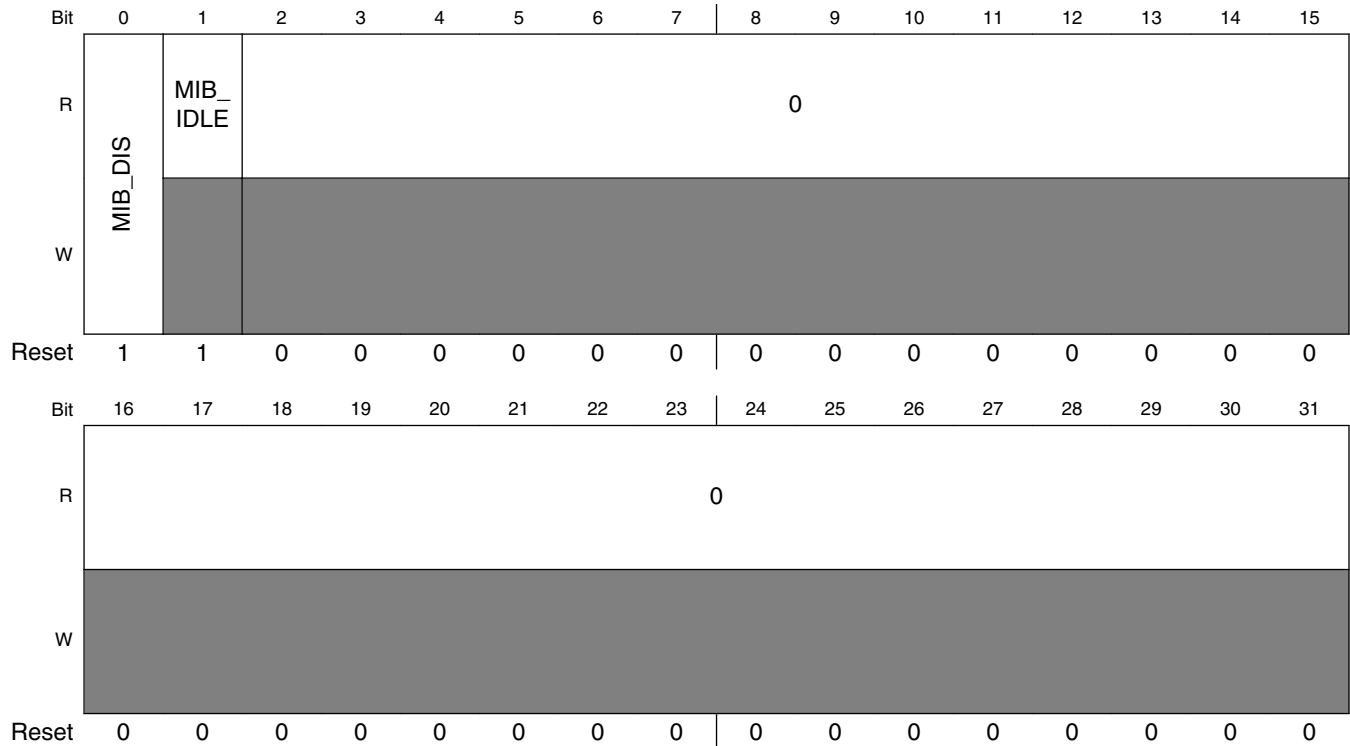
The MIBC is a read/write register controlling and observing the state of the MIB block. User software accesses this register if there is a need to disable the MIB block operation. For example, to clear all MIB counters in RAM:

1. Disable the MIB block

2. Clear all the MIB RAM locations
3. Enable the MIB block

The MIB_DIS bit is reset to 1. See the memory map for the locations of the MIB counters.

Address: 0h base + 64h offset = 64h



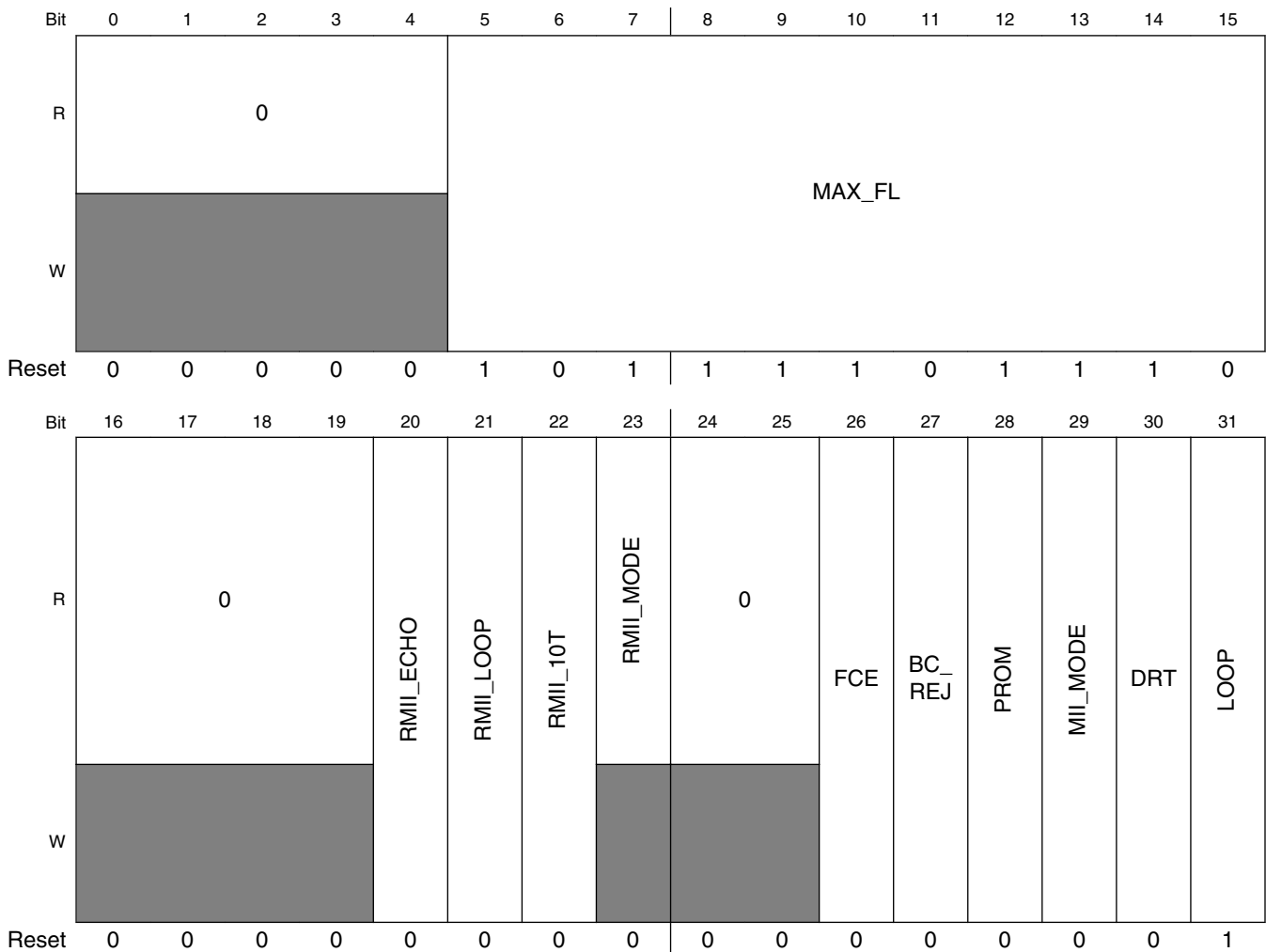
FEC_MIBC field descriptions

Field	Description
0 MIB_DIS	If MIB_DIS = 1, the MIB logic halts and not update any MIB counters.
1 MIB_IDLE	If MIB_IDLE = 1, the MIB block is not currently updating any MIB counters.
2–31 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

46.5.9 Receive Control Register (FEC_RCR)

RCR controls the operational mode of the receive block and must be written only when ECR[ETHER_EN] is cleared (initialization time).

Address: 0h base + 84h offset = 84h



FEC_RCR field descriptions

Field	Description
0-4 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
5-15 MAX_FL	Maximum frame length. Length is measured starting at DA and includes the CRC at the end of the frame. Transmit frames longer than MAX_FL causes the BABT interrupt to occur. Receive frames longer than MAX_FL causes the BABR interrupt to occur and sets the LG bit in the end of frame receive buffer descriptor.

Table continues on the next page...

FEC_RCR field descriptions (continued)

Field	Description
	The recommended default value to be programmed is 1518 or 1522 if VLAN tags are supported.
16–19 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
20 RMII_ECHO	RMII Echo. Enables RMII echo mode. RMII 2-bit receive data is processed through the RMII receive logic to the FEC and also routes through the RMII's transmit logic to become RMII 2-bit transmit data out. 0 Normal operation. 1 RMII echo mode enabled. NOTE: When RMII_ECHO is set, proper operation is guaranteed only when RMII_MODE = 1, RMII_LOOP = 0 and LOOP = 0.
21 RMII_LOOP	RMII loopback. Enables RMII loopback mode. Causes the MII transmit outputs from the Ethernet controller to loop back to the Ethernet controllers's MII receive inputs through the RMII transmit/receive logic. 0 Normal operation. 1 RMII loopback mode enabled. NOTE: When RMII_LOOP is set, proper operation is guaranteed only when RMII_MODE = 1, RMII_ECHO = 0, LOOP = 0, and TCR <i>n</i> [FDEN] = 1.
22 RMII_10T	RMII 10-Base T. Enables 10Mbps mode of the RMII. Determines the clock frequency of the clock source to the FEC logic to support 10/100Mbps operations. 0 100 Mbps operation. The 50 MHz RMII reference clock on FEC_TXCLK is sent to the RMII, while a divided-by-2 version (25 MHz) is sent to the FEC. 1 10 Mbps operation. The 50 MHz RMII reference clock on FEC_TXCLK is divided by 10 (5 MHz) and sent to the RMII, while a divided-by-20 version (2.5 MHz) is sent to the FEC.
23 RMII_MODE	RMII Mode. Indicates if the FEC is in RMII or MII/7-wire mode. See the chip configuration chapter for more details. NOTE: Interface selection is chip-specific; see the chapter that describes how modules are configured and connected. 0 FEC is in true MII mode. 1 FEC is in RMII mode.
24–25 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
26 FCE	Flow control enable. If asserted, the receiver detects PAUSE frames. Upon PAUSE frame detection, the transmitter stops transmitting data frames for a given duration.
27 BC_REJ	Broadcast frame reject. If asserted, frames with DA (destination address) equal to FFFF_FFFF_FFFF are rejected unless the PROM bit is set. If BC_REJ and PROM are set, frames with broadcast DA are accepted and the M (MISS) is set in the receive buffer descriptor.
28 PROM	Promiscuous mode. All frames are accepted regardless of address matching.
29 MII_MODE	Media independent interface mode. Selects the external interface mode for transmit and receive blocks. NOTE: Interface selection is chip-specific; see the chapter that describes how modules are configured and connected. 0 7-wire mode (used only for serial 10 Mbps) 1 MII or RMII mode as indicated by the RMII_MODE bit

Table continues on the next page...

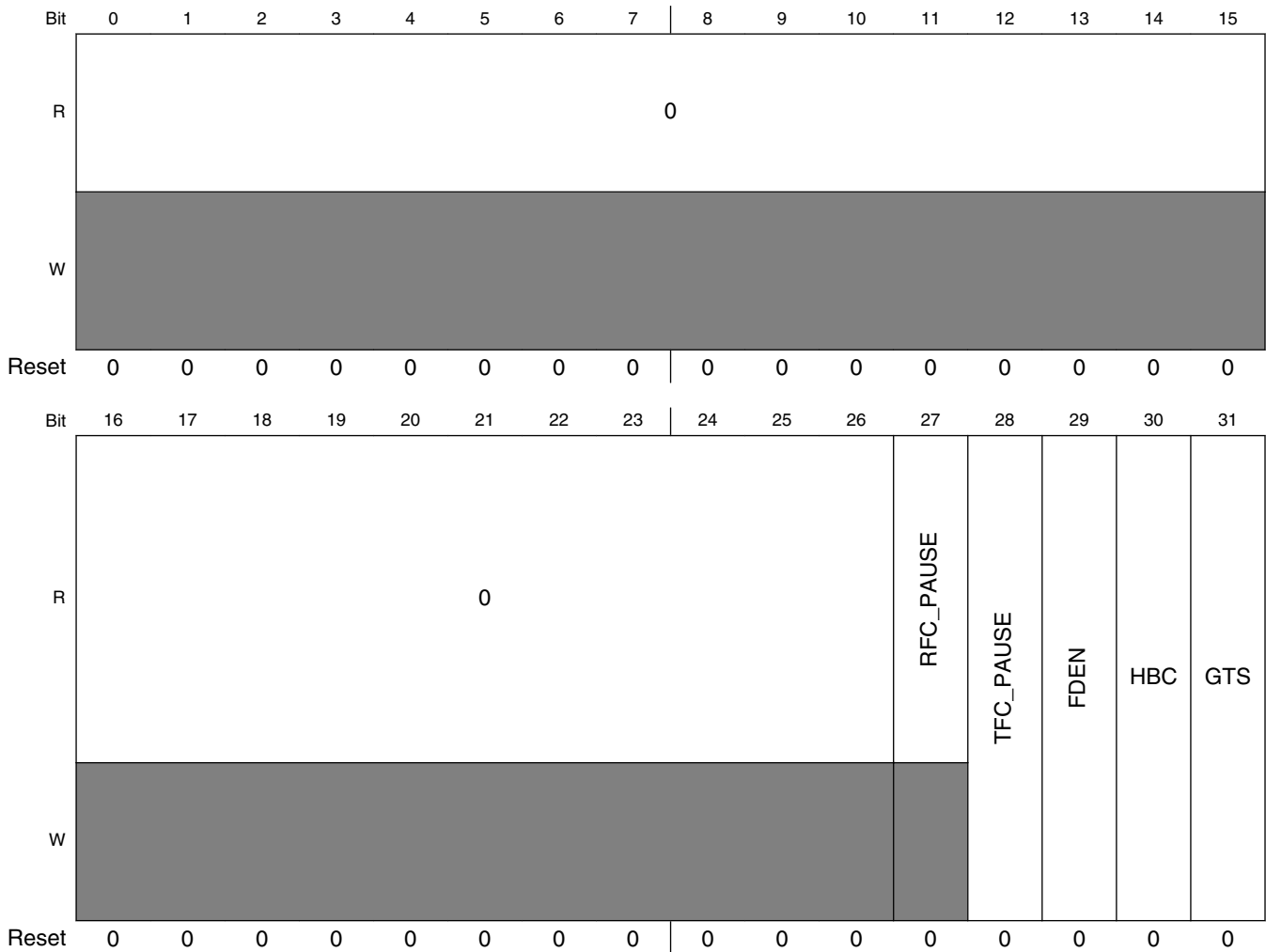
FEC_RCR field descriptions (continued)

Field	Description
30 DRT	Disable receive on transmit. 0 Receive path operates independently of transmit (use for full duplex or to monitor transmit activity in half duplex mode). 1 Disable reception of frames while transmitting (normally used for half duplex mode).
31 LOOP	Internal loopback. If set, transmitted frames are looped back internal to the device and transmit output signals are not asserted. The internal bus clock substitutes for the FEC_TXCLK when LOOP is asserted. DRT must be set to 0 when setting LOOP.

46.5.10 Transmit Control Register (FEC_TCR)

The TCR configures the transmit block.

Address: 0h base + C4h offset = C4h



FEC_TCR field descriptions

Field	Description
0–26 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
27 RFC_PAUSE	Receive frame control pause. This read-only status bit is asserted when a full duplex flow control pause frame is received and the transmitter pauses for the duration defined in this pause frame. This bit automatically clears when the pause duration is complete.
28 TFC_PAUSE	Transmit frame control pause. Transmits a PAUSE frame when asserted. When this bit is set, the MAC stops transmission of data frames after the current transmission is complete. At this time, GRA interrupt in the EIR is asserted. With transmission of data frames stopped, MAC transmits a MAC Control PAUSE frame. Next, the MAC clears the TFC_PAUSE bit and resumes transmitting data frames. If the transmitter pauses due to user assertion of GTS or reception of a PAUSE frame, the MAC may continue transmitting a MAC Control PAUSE frame.
29 FDEN	Full duplex enable. If set, frames transmit independent of carrier sense and collision inputs. This field must be modified only when ECR[ETHER_EN] = 0.
30 HBC	Heartbeat control. If set, the heartbeat check performs following end of transmission and the HB bit in the status register is set if the collision input does not assert within the heartbeat window. This field must be modified only when ECR[ETHER_EN] = 0.
31 GTS	Graceful transmit stop. When this bit is set, MAC stops transmission after any frame currently transmitted is complete and GRA interrupt in the EIR is asserted. If frame transmission is not currently underway, the GRA interrupt is asserted immediately. After transmission finishes, clear GTS to restart. The next frame in the transmit FIFO is then transmitted. If an early collision occurs during transmission when GTS is set, transmission stops after the collision. The frame is transmitted again after GTS is cleared. There may be old frames in the transmit FIFO that transmit when GTS is reasserted. To avoid this situation, program ECR[ETHER_EN] = 0 following the GRA interrupt.

46.5.11 Physical Address Low Register (FEC_PALR)

PALR contains the lower 32 bits (bytes 0, 1, 2, 3) of the 48-bit address used in the address recognition process to compare with the DA (destination address) field of receive frames with an individual DA. In addition, this register is used in bytes 0 through 3 of the 6-byte source address field when transmitting PAUSE frames.

This register is not reset and you must initialize it.

Address: 0h base + E4h offset = E4h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	PADDR1																															
W																																
Reset	u*	u*	u*	u*	u*	u*	u*	u*	u*	u*	u*	u*	u*	u*	u*	u*	u*	u*	u*	u*	u*	u*	u*	u*	u*	u*	u*	u*	u*	u*	u*	u*

* Notes:

- u = Unaffected by reset.

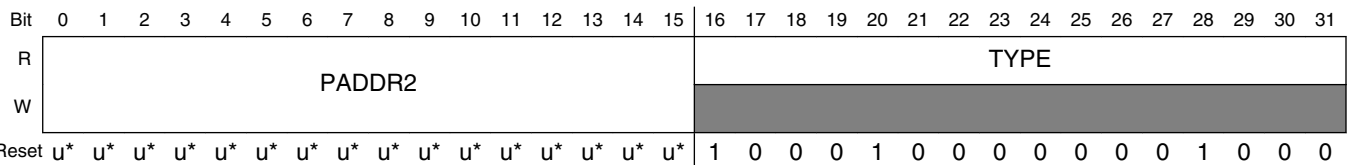
FEC_PALR field descriptions

Field	Description
0–31 PADDR1	Bytes 0 (bits 31:24), 1 (bits 23:16), 2 (bits 15:8), and 3 (bits 7:0) of the 6-byte individual address are used for exact match and the source address field in PAUSE frames.

46.5.12 Physical Address High Register and Type Field (FEC_PAUR)

PAUR contains the upper 16 bits (bytes 4 and 5) of the 48-bit address used in the address recognition process to compare with the DA (destination address) field of receive frames with an individual DA. In addition, this register is used in bytes 4 and 5 of the 6-byte Source Address field when transmitting PAUSE frames. Bits 15:0 of PAUR contain a constant type field (8808h) for transmission of PAUSE frames. The upper 16 bits of this register are not reset and you must initialize them.

Address: 0h base + E8h offset = E8h



- * Notes:
- u = Unaffected by reset.

FEC_PAUR field descriptions

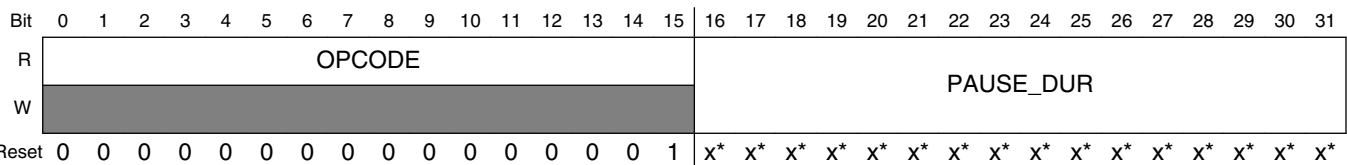
Field	Description
0–15 PADDR2	Bytes 4 (bits 31:24) and 5 (bits 23:16) of the 6-byte individual address used for exact match, and the source address field in PAUSE frames.
16–31 TYPE	Type field in PAUSE frames. These 16 read-only bits are a constant value of 8808h.

46.5.13 Opcode/Pause Duration (FEC_OPD)

The OPD contains the 16-bit opcode and 16-bit pause duration fields used in transmission of a PAUSE frame. The opcode field is a constant value, 0001h. When another node detects a PAUSE frame, that node pauses transmission for the duration specified in the pause duration field.

The lower 16 bits of this register are not reset and you must initialize them.

Address: 0h base + ECh offset = ECh



* Notes:

- x = Undefined at reset.

FEC_OPD field descriptions

Field	Description
0–15 OPCODE	Opcode field used in PAUSE frames. These read-only bits are a constant, 0001h.
16–31 PAUSE_DUR	Pause Duration field used in PAUSE frames.

46.5.14 Descriptor Individual Upper Address Register (FEC_IAUR)

IAUR contains the upper 32 bits of the 64-bit individual address hash table. The address recognition process uses this table to check for a possible match with the destination address (DA) field of receive frames with an individual DA.

This register is not reset and you must initialize it.

Address: 0h base + 118h offset = 118h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R																																
W	IADDR1																															
Reset	u*	u*	u*	u*	u*	u*	u*	u*	u*	u*	u*	u*	u*	u*	u*	u*	u*	u*	u*	u*	u*	u*	u*	u*	u*	u*	u*	u*	u*	u*	u*	

* Notes:

- u = Unaffected by reset.

FEC_IAUR field descriptions

Field	Description
0–31 IADDR1	The upper 32 bits of the 64-bit hash table used in the address recognition process for receive frames with a unicast address. Bit 31 of IADDR1 contains hash index bit 63. Bit 0 of IADDR1 contains hash index bit 32.

46.5.15 Descriptor Individual Lower Address Register (FEC_IALR)

IALR contains the lower 32 bits of the 64-bit individual address hash table. The address recognition process uses this table to check for a possible match with the DA field of receive frames with an individual DA.

This register is not reset and you must initialize it.

Memory map and register definition

Address: 0h base + 11Ch offset = 11Ch

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	IADDR2																															
W																																
Reset	u*	u*	u*	u*	u*	u*	u*	u*	u*	u*	u*	u*	u*	u*	u*	u*	u*	u*	u*	u*	u*	u*	u*	u*	u*	u*	u*	u*	u*	u*	u*	

* Notes:

- u = Unaffected by reset.

FEC_IALR field descriptions

Field	Description
0–31 IADDR2	The lower 32 bits of the 64-bit hash table used in the address recognition process for receive frames with a unicast address. Bit 31 of IADDR2 contains hash index bit 31. Bit 0 of IADDR2 contains hash index bit 0.

46.5.16 Descriptor Group Upper Address Register (FEC_GAUR)

GAUR contains the upper 32 bits of the 64-bit hash table used in the address recognition process for receive frames with a multicast address.

This register is not reset and you must initialize it.

Address: 0h base + 120h offset = 120h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	GADDR1																															
W																																
Reset	u*	u*	u*	u*	u*	u*	u*	u*	u*	u*	u*	u*	u*	u*	u*	u*	u*	u*	u*	u*	u*	u*	u*	u*	u*	u*	u*	u*	u*	u*	u*	

* Notes:

- u = Unaffected by reset.

FEC_GAUR field descriptions

Field	Description
0–31 GADDR1	The GADDR1 register contains the upper 32 bits of the 64-bit hash table used in the address recognition process for receive frames with a multicast address. Bit 31 of GADDR1 contains hash index bit 63. Bit 0 of GADDR1 contains hash index bit 32.

46.5.17 Descriptor Group Lower Address Register (FEC_GALR)

GALR contains the lower 32 bits of the 64-bit hash table used in the address recognition process for receive frames with a multicast address.

This register is not reset and you must initialize it.

Address: 0h base + 124h offset = 124h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R																																
W																																
Reset	u*	u*	u*	u*	u*	u*	u*	u*	u*	u*	u*	u*	u*	u*	u*	u*	u*	u*	u*	u*	u*	u*	u*	u*	u*	u*	u*	u*	u*	u*	u*	u*

* Notes:

- u = Unaffected by reset.

FEC_GALR field descriptions

Field	Description
0–31 GADDR2	The GADDR2 register contains the lower 32 bits of the 64-bit hash table used in the address recognition process for receive frames with a multicast address. Bit 31 of GADDR2 contains hash index bit 31. Bit 0 of GADDR2 contains hash index bit 0.

46.5.18 Transmit FIFO Watermark (FEC_TFWR)

The TFWR controls the amount of data required in the transmit FIFO before transmission of a frame can begin. This allows you to minimize transmit latency (TFWR = 00 or 01) or allow for larger bus access latency (TFWR = 11) due to contention for the system bus. Setting the watermark to a high value minimizes the risk of transmit FIFO underrun due to contention for the system bus. The byte counts associated with the TFWR field may need to be modified to match a given system requirement (worst case bus access latency by the transmit data DMA channel).

Address: 0h base + 144h offset = 144h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0														TFWR	
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

FEC_TFWR field descriptions

Field	Description
0–29 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
30–31 TFWR	Number of bytes written to transmit FIFO before transmission of a frame begins 00 64 bytes written

Table continues on the next page...

FEC_TFWR field descriptions (continued)

Field	Description
01	64 bytes written
10	128 bytes written
11	192 bytes written

46.5.19 FIFO Receive Bound Register (FEC_FRBR)

FRBR indicates the upper address bound of the FIFO RAM. Drivers can use this value, along with the FRSR, to appropriately divide the available FIFO RAM between the transmit and receive data paths.

Address: 0h base + 14Ch offset = 14Ch

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15		
R	0																	
W																		
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0		
Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31		
R	0					1	R_BOUND										0	
W																		
Reset	0	0	0	0	0	1	1	0	0	0	0	0	0	0	0	0		

FEC_FRBR field descriptions

Field	Description
0–20 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
21 Reserved	This field is reserved. This read-only field is reserved and always has the value 1.
22–29 R_BOUND	Highest valid FIFO RAM address.
30–31 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

46.5.20 FIFO Receive Start Register (FEC_FRSR)

FRSR indicates the starting address of the receive FIFO. FRSR marks the boundary between the transmit and receive FIFOs. The transmit FIFO uses addresses from the start of the FIFO to the location four bytes before the address programmed into the FRSR. The receive FIFO uses addresses from FRSR to FRBR inclusive.

Hardware initializes the FRSR at reset. FRSR only needs to be written to change the default value.

Address: 0h base + 150h offset = 150h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31		
R	0					1	R_FSTART										0	
W							R_FSTART											
Reset	0	0	0	0	0	1	0	1	0	0	0	0	0	0	0	0		

FEC_FRSR field descriptions

Field	Description
0–20 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
21 Reserved	This field is reserved. This read-only field is reserved and always has the value 1.
22–29 R_FSTART	Address of first receive FIFO location. Acts as delimiter between receive and transmit FIFOs. For proper operation, ensure that R_FSTART is set to 48h or greater.
30–31 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

46.5.21 Receive Descriptor Ring Start Register (FEC_ERDSR)

ERDSR points to the start of the circular receive buffer descriptor queue in external memory. This pointer must be 32-bit aligned; however, it is recommended it be made 128-bit aligned (evenly divisible by 16). You should write zeros to bits 1 and 0. Hardware ignores non-zero values in these two bit positions.

This register is undefined at reset and must be initialized prior to operation.

Address: 0h base + 180h offset = 180h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	R_DES_START																														0	
W																																
Reset	u*	u*	u*	u*	u*	u*	u*	u*	u*	u*	u*	u*	u*	u*	u*	u*	u*	u*	u*	u*	u*	u*	u*	u*	u*	u*	u*	u*	u*	u*	u*	u*

* Notes:

- u = Unaffected by reset.

FEC_ERDSR field descriptions

Field	Description
0–29 R_DES_START	Pointer to start of receive buffer descriptor queue.
30–31 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

46.5.22 Transmit Buffer Descriptor Ring Start Register (FEC_ETDSR)

ETSDR provides a pointer to the start of the circular transmit buffer descriptor queue in external memory. This pointer must be 32-bit aligned; however, it is recommended it be made 128-bit aligned (evenly divisible by 16). You should write zeros to bits 1 and 0. Hardware ignores non-zero values in these two bit positions.

This register is undefined at reset and must be initialized prior to operation.

Address: 0h base + 184h offset = 184h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	X_DES_START																														0	
W																																
Reset	u*	u*	u*	u*	u*	u*	u*	u*	u*	u*	u*	u*	u*	u*	u*	u*	u*	u*	u*	u*	u*	u*	u*	u*	u*	u*	u*	u*	u*	u*	u*	

* Notes:

- u = Unaffected by reset.

FEC_ETDSR field descriptions

Field	Description
0–29 X_DES_START	Pointer to start of transmit buffer descriptor queue.
30–31 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

46.5.23 Receive Buffer Size Register (FEC_EMRBR)

The EMRBR dictates the maximum size of all receive buffers. This value should take into consideration that the receive CRC is always written into the last receive buffer. To allow one maximum size frame per buffer, EMRBR must be set to RCR[MAX_FL] or larger. To properly align the buffer, EMRBR must be evenly divisible by 16. To ensure this, bits 3-0 are forced low.

To minimize bus utilization (descriptor fetches), it is recommended that EMRBR be greater than or equal to 256 bytes.

The EMRBR is undefined at reset and must be initialized by the user.

Address: 0h base + 188h offset = 188h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0																				R_BUF_SIZE								0			
W																																
Reset	u*	u*	u*	u*	u*	u*	u*	u*	u*	u*	u*	u*	u*	u*	u*	u*	u*	u*	u*	u*	u*	u*	u*	u*	u*	u*	u*	u*	u*	u*	u*	u*

* Notes:

- u = Unaffected by reset.

FEC_EMRBR field descriptions

Field	Description
0–20 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
21–27 R_BUF_SIZE	Maximum size of receive buffer size in bytes. To minimize bus utilization (descriptor fetches), set this field to 256 bytes (10h) or larger. 10h: 256 + 15 bytes (minimum size recommended) 11h: 272 + 15 bytes ... 7Fh: 2032 + 15 bytes. The FEC writes up to 2047 bytes in the receive buffer. If data larger than 2047 is received, the FEC truncates it and shows 7FFh in the receive descriptor
28–31 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

46.5.24 Count of frames not counted correctly (FEC_RMON_T_DROP)

See [Using the RMON and IEEE registers](#).

Address: 0h base + 200h offset = 200h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R																																
W	RMON_T_DROP																															
Reset	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*

* Notes:

- x = Undefined at reset.

FEC_RMON_T_DROP field descriptions

Field	Description
0–31 RMON_T_DROP	Count of frames not counted correctly

46.5.25 RMON Tx packet count (FEC_RMON_T_PACKETS)

See [Using the RMON and IEEE registers](#).

Address: 0h base + 204h offset = 204h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	RMON_T_PACKETS																															
W																																
Reset	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	

* Notes:

- x = Undefined at reset.

FEC_RMON_T_PACKETS field descriptions

Field	Description
0–31 RMON_T_PACKETS	RMON Tx packet count

46.5.26 RMON Tx broadcast packets (FEC_RMON_T_BC_PKT)

See [Using the RMON and IEEE registers](#).

Address: 0h base + 208h offset = 208h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	RMON_T_BC_PKT																															
W																																
Reset	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	

* Notes:

- x = Undefined at reset.

FEC_RMON_T_BC_PKT field descriptions

Field	Description
0–31 RMON_T_BC_PKT	RMON Tx broadcast packets

46.5.27 RMON Tx multicast packets (FEC_RMONT_MC_PKT)

See [Using the RMON and IEEE registers](#).

Address: 0h base + 20Ch offset = 20Ch

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	RMON_T_MC_PKT																															
W																																
Reset	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	

* Notes:

- x = Undefined at reset.

FEC_RMONT_MC_PKT field descriptions

Field	Description
0–31 RMONT_MC_PKT	RMONT Tx multicast packets

46.5.28 RMON Tx packets with CRC/align error (FEC_RMONT_CRC_ALIGN)

See [Using the RMON and IEEE registers](#).

Address: 0h base + 210h offset = 210h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	RMON_T_CRC_ALIGN																															
W																																
Reset	X*	X*	X*	X*	X*	X*	X*	X*	X*	X*	X*	X*	X*	X*	X*	X*	X*	X*	X*	X*	X*	X*	X*	X*	X*	X*	X*	X*	X*	X*	X*	

* Notes:

- x = Undefined at reset.

FEC_RMONT_CRC_ALIGN field descriptions

Field	Description
0–31 RMONT_CRC_ALIGN	RMONT Tx packets with CRC/align error

46.5.29 RMON Tx packets < 64 bytes, good CRC (FEC_RMONT_UNDERSIZE)

See [Using the RMON and IEEE registers](#).

Address: 0h base + 214h offset = 214h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	RMON_T_UNDERSIZE																															
W																																
Reset	X*	X*	X*	X*	X*	X*	X*	X*	X*	X*	X*	X*	X*	X*	X*	X*	X*	X*	X*	X*	X*	X*	X*	X*	X*	X*	X*	X*	X*	X*	X*	

* Notes:

- x = Undefined at reset.

FEC_RMONT_UNDERSIZE field descriptions

Field	Description
0–31 RMONT_UNDERSIZE	RMON Tx packets < 64 bytes, good CRC

46.5.30 RMON Tx packets > MAX_FL bytes, good CRC (FEC_RMONT_OVERSIZE)

See [Using the RMON and IEEE registers](#).

Address: 0h base + 218h offset = 218h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	RMON_T_OVERSIZE																															
W																																
Reset	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	

* Notes:

- x = Undefined at reset.

FEC_RMONT_OVERSIZE field descriptions

Field	Description
0–31 RMONT_OVERSIZE	RMON Tx packets > MAX_FL bytes, good CRC

46.5.31 RMON Tx packets < 64 bytes, bad CRC (FEC_RMONT_FRAG)

See [Using the RMON and IEEE registers](#).

Address: 0h base + 21Ch offset = 21Ch

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	RMON_T_FRAG																															
W																																
Reset	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	

* Notes:

- x = Undefined at reset.

FEC_RMONT_FRAG field descriptions

Field	Description
0–31 RMONT_FRAG	RMON Tx packets < 64 bytes, bad CRC

46.5.32 RMON Tx packets > MAX_FL bytes, bad CRC (FEC_RMONT_JAB)

See [Using the RMON and IEEE registers](#).

Address: 0h base + 220h offset = 220h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	RMON_T_JAB																															
W																																
Reset	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	

* Notes:

- x = Undefined at reset.

FEC_RMONT_JAB field descriptions

Field	Description
0–31 RMONT_JAB	RMON Tx packets > MAX_FL bytes, bad CRC

46.5.33 RMON Tx collision count (FEC_RMONT_COL)

See [Using the RMON and IEEE registers](#).

Address: 0h base + 224h offset = 224h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	RMON_T_COL																															
W																																
Reset	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	

* Notes:

- x = Undefined at reset.

FEC_RMONT_COL field descriptions

Field	Description
0–31 RMONT_COL	RMON Tx collision count

46.5.34 RMON Tx 64 byte packets (FEC_RMONT_P64)

See [Using the RMON and IEEE registers](#).

Address: 0h base + 228h offset = 228h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	RMON_T_P64																															
W																																
Reset	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	

* Notes:

- x = Undefined at reset.

FEC_RMONT_P64 field descriptions

Field	Description
0–31 RMONT_P64	RMON Tx 64 byte packets

46.5.35 RMON Tx 65 to 127 byte packets (FEC_RMONT_P65TO127)

See [Using the RMON and IEEE registers](#).

Address: 0h base + 22Ch offset = 22Ch

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	RMON_T_P65TO127																															
W																																
Reset	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	

* Notes:

- x = Undefined at reset.

FEC_RMONT_P65TO127 field descriptions

Field	Description
0–31 RMONT_P65TO127	RMON Tx 65 to 127 byte packets

46.5.36 RMON Tx 128 to 255 byte packets (FEC_RMONT_P128TO255)

See [Using the RMON and IEEE registers](#).

Address: 0h base + 230h offset = 230h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	RMON_T_P128TO255																															
W																																
Reset	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	

* Notes:

- x = Undefined at reset.

FEC_RMONT_P128TO255 field descriptions

Field	Description
0–31 RMONT_P128TO255	RMON Tx 128 to 255 byte packets

46.5.37 RMON Tx 256 to 511 byte packets (FEC_RMONT_P256TO511)

See [Using the RMON and IEEE registers](#).

Address: 0h base + 234h offset = 234h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	RMON_T_P256TO511																															
W																																
Reset	X*	X*	X*	X*	X*	X*	X*	X*	X*	X*	X*	X*	X*	X*	X*	X*	X*	X*	X*	X*	X*	X*	X*	X*	X*	X*	X*	X*	X*	X*	X*	

* Notes:

- x = Undefined at reset.

FEC_RMONT_P256TO511 field descriptions

Field	Description
0–31 RMONT_P256TO511	RMON Tx 256 to 511 byte packets

46.5.38 RMON Tx 512 to 1023 byte packets (FEC_RMONT_P512TO1023)

See [Using the RMON and IEEE registers](#).

Address: 0h base + 238h offset = 238h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	RMON_T_P512TO1023																															
W																																
Reset	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	

* Notes:

- x = Undefined at reset.

FEC_RMONT_P512TO1023 field descriptions

Field	Description
0–31 RMONT_P512TO1023	RMON Tx 512 to 1023 byte packets

46.5.39 RMON Tx 1024 to 2047 byte packets (FEC_RMONT_P1024TO2047)

See [Using the RMON and IEEE registers](#).

Address: 0h base + 23Ch offset = 23Ch

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	RMON_T_P1024TO2047																															
W																																
Reset	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	

* Notes:

- x = Undefined at reset.

FEC_RMONT_P1024TO2047 field descriptions

Field	Description
0–31 RMONT_P1024TO2047	RMON Tx 1024 to 2047 byte packets

46.5.40 RMON Tx packets with > 2048 bytes (FEC_RMONT_P_GTE2048)

See [Using the RMON and IEEE registers](#).

Address: 0h base + 240h offset = 240h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	RMON_T_P_GTE2048																															
W																																
Reset	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	

* Notes:

- x = Undefined at reset.

FEC_RMONT_P_GTE2048 field descriptions

Field	Description
0–31 RMONT_P_GTE2048	RMON Tx packets with > 2048 bytes

46.5.41 RMON Tx Octets (FEC_RMON_T_OCTETS)

See [Using the RMON and IEEE registers](#).

Address: 0h base + 244h offset = 244h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	RMON_T_OCTETS																															
W																																
Reset	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	

* Notes:

- x = Undefined at reset.

FEC_RMON_T_OCTETS field descriptions

Field	Description
0–31 RMON_T_OCTETS	RMON Tx Octets

46.5.42 Count of transmitted frames not counted correctly (FEC_IEEE_T_DROP)

See [Using the RMON and IEEE registers](#).

Address: 0h base + 248h offset = 248h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	IEEE_T_DROP																															
W																																
Reset	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	

* Notes:

- x = Undefined at reset.

FEC_IEEE_T_DROP field descriptions

Field	Description
0–31 IEEE_T_DROP	Count of transmitted frames not counted correctly

46.5.43 Frames transmitted OK (FEC_IEEE_T_FRAME_OK)

See [Using the RMON and IEEE registers](#).

Address: 0h base + 24Ch offset = 24Ch

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	IEEE_T_FRAME_OK																															
W																																
Reset	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	

* Notes:

- x = Undefined at reset.

FEC_IEEE_T_FRAME_OK field descriptions

Field	Description
0–31 IEEE_T_FRAME_OK	Frames transmitted OK

46.5.44 Frames transmitted with single collision (FEC_IEEE_T_1COL)

See [Using the RMON and IEEE registers](#).

Address: 0h base + 250h offset = 250h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	IEEE_T_1COL																															
W																																
Reset	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	

* Notes:

- x = Undefined at reset.

FEC_IEEE_T_1COL field descriptions

Field	Description
0–31 IEEE_T_1COL	Frames transmitted with single collision

46.5.45 Frames transmitted with multiple collisions (FEC_IEEE_T_MCOL)

See [Using the RMON and IEEE registers](#).

Address: 0h base + 254h offset = 254h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	IEEE_T_MCOL																															
W																																
Reset	X*	X*	X*	X*	X*	X*	X*	X*	X*	X*	X*	X*	X*	X*	X*	X*	X*	X*	X*	X*	X*	X*	X*	X*	X*	X*	X*	X*	X*	X*	X*	

* Notes:

- x = Undefined at reset.

FEC_IEEE_T_MCOL field descriptions

Field	Description
0–31 IEEE_T_MCOL	Frames transmitted with multiple collisions

46.5.46 Frames transmitted after deferral delay (FEC_IEEE_T_DEF)

See [Using the RMON and IEEE registers](#).

Address: 0h base + 258h offset = 258h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	IEEE_T_DEF																															
W																																
Reset	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	

* Notes:

- x = Undefined at reset.

FEC_IEEE_T_DEF field descriptions

Field	Description
0–31 IEEE_T_DEF	Frames transmitted after deferral delay

46.5.47 Frames transmitted with late collision (FEC_IEEE_T_LCOL)

See [Using the RMON and IEEE registers](#).

Address: 0h base + 25Ch offset = 25Ch

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	IEEE_T_LCOL																															
W																																
Reset	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	

* Notes:

- x = Undefined at reset.

FEC_IEEE_T_LCOL field descriptions

Field	Description
0–31 IEEE_T_LCOL	Frames transmitted with late collision

46.5.48 Frames transmitted with excessive collisions (FEC_IEEE_T_EXCOL)

See [Using the RMON and IEEE registers](#).

Address: 0h base + 260h offset = 260h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	IEEE_T_EXCOL																															
W																																
Reset	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	

* Notes:

- x = Undefined at reset.

FEC_IEEE_T_EXCOL field descriptions

Field	Description
0–31 IEEE_T_EXCOL	Frames transmitted with excessive collisions

46.5.49 Frames transmitted with Tx FIFO underrun (FEC_IEEE_T_MACERR)

See [Using the RMON and IEEE registers](#).

Address: 0h base + 264h offset = 264h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	IEEE_T_MACERR																															
W																																
Reset	X*	X*	X*	X*	X*	X*	X*	X*	X*	X*	X*	X*	X*	X*	X*	X*	X*	X*	X*	X*	X*	X*	X*	X*	X*	X*	X*	X*	X*	X*	X*	

* Notes:

- x = Undefined at reset.

FEC_IEEE_T_MACERR field descriptions

Field	Description
0–31 IEEE_T_MACERR	Frames transmitted with Tx FIFO underrun

46.5.50 Frames transmitted with carrier sense error (FEC_IEEE_T_CSERR)

See [Using the RMON and IEEE registers](#).

Address: 0h base + 268h offset = 268h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	IEEE_T_CSERR																															
W																																
Reset	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	

* Notes:

- x = Undefined at reset.

FEC_IEEE_T_CSERR field descriptions

Field	Description
0–31 IEEE_T_CSERR	Frames transmitted with carrier sense error

46.5.51 Frames transmitted with SQE error (FEC_IEEE_T_SQE)

See [Using the RMON and IEEE registers](#).

Address: 0h base + 26Ch offset = 26Ch

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	IEEE_T_SQE																															
W																																
Reset	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*

* Notes:

- x = Undefined at reset.

FEC_IEEE_T_SQE field descriptions

Field	Description
0–31 IEEE_T_SQE	Frames transmitted with SQE error

46.5.52 Flow control pause frames transmitted (FEC_IEEE_T_FDXFC)

See [Using the RMON and IEEE registers](#).

Address: 0h base + 270h offset = 270h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	IEEE_T_FDXFC																															
W																																
Reset	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*

* Notes:

- x = Undefined at reset.

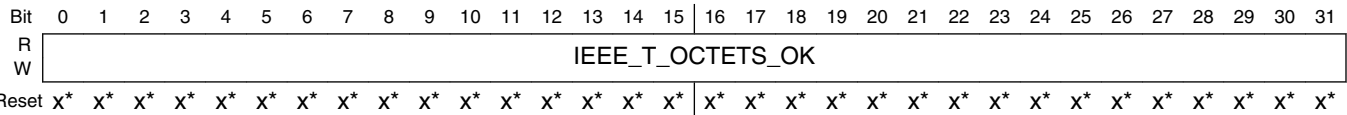
FEC_IEEE_T_FDXFC field descriptions

Field	Description
0–31 IEEE_T_FDXFC	Flow control pause frames transmitted

46.5.53 **Octet count for frames transmitted without error
(FEC_IEEE_T_OCTETS_OK)**

See [Using the RMON and IEEE registers](#).

Address: 0h base + 274h offset = 274h



- * Notes:
- x = Undefined at reset.

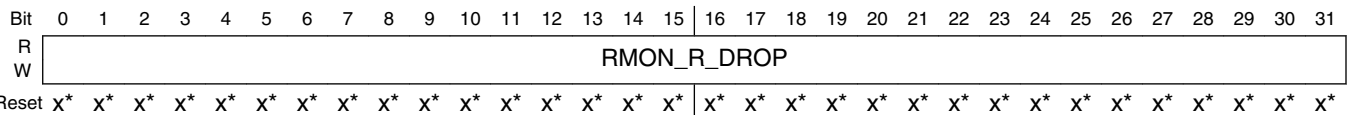
FEC_IEEE_T_OCTETS_OK field descriptions

Field	Description
0–31 IEEE_T_OCTETS_OK	Octet count for frames transmitted without error

46.5.54 **Count of received frames not counted correctly
(FEC_RMON_R_DROP)**

See [Using the RMON and IEEE registers](#).

Address: 0h base + 280h offset = 280h



- * Notes:
- x = Undefined at reset.

FEC_RMON_R_DROP field descriptions

Field	Description
0–31 RMON_R_DROP	Count of received frames not counted correctly

46.5.55 RMON Rx packet count (FEC_RMON_R_PACKETS)

See [Using the RMON and IEEE registers](#).

Address: 0h base + 284h offset = 284h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	RMON_R_PACKETS																															
W																																
Reset	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	

* Notes:

- x = Undefined at reset.

FEC_RMON_R_PACKETS field descriptions

Field	Description
0–31 RMON_R_PACKETS	RMON Rx packet count

46.5.56 RMON Rx broadcast packets (FEC_RMON_R_BC_PKT)

See [Using the RMON and IEEE registers](#).

Address: 0h base + 288h offset = 288h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	RMON_R_BC_PKT																															
W																																
Reset	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	

* Notes:

- x = Undefined at reset.

FEC_RMON_R_BC_PKT field descriptions

Field	Description
0–31 RMON_R_BC_PKT	RMON Rx broadcast packets

46.5.57 RMON Rx multicast packets (FEC_RMON_R_MC_PKT)

See [Using the RMON and IEEE registers](#).

Address: 0h base + 28Ch offset = 28Ch

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	RMON_R_MC_PKT																															
W																																
Reset	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	

* Notes:

- x = Undefined at reset.

FEC_RMON_R_MC_PKT field descriptions

Field	Description
0–31 RMON_R_MC_PKT	RMON Rx multicast packets

46.5.58 RMON Rx packets with CRC/Align error (FEC_RMON_R_CRC_ALIGN)

See [Using the RMON and IEEE registers](#).

Address: 0h base + 290h offset = 290h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	RMON_R_CRC_ALIGN																															
W																																
Reset	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	

* Notes:

- x = Undefined at reset.

FEC_RMON_R_CRC_ALIGN field descriptions

Field	Description
0–31 RMON_R_CRC_ALIGN	RMON Rx packets with CRC/Align error

46.5.59 RMON Rx packets < 64 bytes, good CRC (FEC_RMONT_R_UNDERSIZE)

See [Using the RMON and IEEE registers](#).

Address: 0h base + 294h offset = 294h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	RMON_R_UNDERSIZE																															
W																																
Reset	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	

* Notes:

- x = Undefined at reset.

FEC_RMONT_R_UNDERSIZE field descriptions

Field	Description
0–31 RMONT_R_UNDERSIZE	RMONT Rx packets < 64 bytes, good CRC

46.5.60 RMON Rx packets > MAX_FL bytes, good CRC (FEC_RMONT_R_OVERSIZE)

See [Using the RMON and IEEE registers](#).

Address: 0h base + 298h offset = 298h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	RMON_R_OVERSIZE																															
W																																
Reset	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	

* Notes:

- x = Undefined at reset.

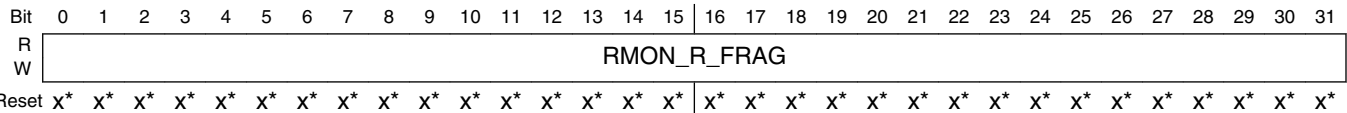
FEC_RMONT_R_OVERSIZE field descriptions

Field	Description
0–31 RMONT_R_OVERSIZE	RMONT Rx packets > MAX_FL bytes, good CRC

46.5.61 RMON Rx packets < 64 bytes, bad CRC (FEC_RMON_R_FRAG)

See [Using the RMON and IEEE registers](#).

Address: 0h base + 29Ch offset = 29Ch



- * Notes:
- x = Undefined at reset.

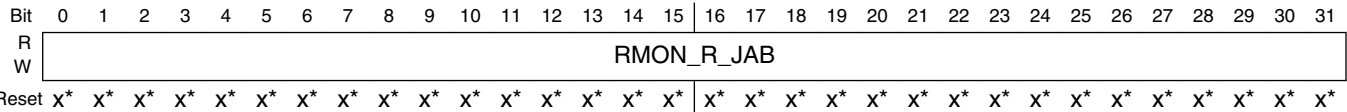
FEC_RMON_R_FRAG field descriptions

Field	Description
0–31 RMON_R_FRAG	RMON Rx packets < 64 bytes, bad CRC

46.5.62 RMON Rx packets > MAX_FL bytes, bad CRC (FEC_RMON_R_JAB)

See [Using the RMON and IEEE registers](#).

Address: 0h base + 2A0h offset = 2A0h



- * Notes:
- x = Undefined at reset.

FEC_RMON_R_JAB field descriptions

Field	Description
0–31 RMON_R_JAB	RMON Rx packets > MAX_FL bytes, bad CRC

46.5.63 Reserved (FEC_RMON_R_RESVD_0)

See [Using the RMON and IEEE registers](#).

Address: 0h base + 2A4h offset = 2A4h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	RMON_R_RESVD_0																															
W																																
Reset	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	

* Notes:

- x = Undefined at reset.

FEC_RMON_R_RESVD_0 field descriptions

Field	Description
0–31 RMON_R_ RESVD_0	Reserved

46.5.64 RMON Rx 64 byte packets (FEC_RMON_R_P64)

See [Using the RMON and IEEE registers](#).

Address: 0h base + 2A8h offset = 2A8h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	RMON_R_P64																															
W																																
Reset	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	

* Notes:

- x = Undefined at reset.

FEC_RMON_R_P64 field descriptions

Field	Description
0–31 RMON_R_P64	RMON Rx 64 byte packets

46.5.65 RMON Rx 65 to 127 byte packets (FEC_RMON_R_P65TO127)

See [Using the RMON and IEEE registers](#).

Address: 0h base + 2ACh offset = 2ACh

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	RMON_R_P65TO127																															
W																																
Reset	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	

* Notes:

- x = Undefined at reset.

FEC_RMON_R_P65TO127 field descriptions

Field	Description
0–31 RMON_R_ P65TO127	RMON Rx 65 to 127 byte packets

46.5.66 RMON Rx 128 to 255 byte packets (FEC_RMON_R_P128TO255)

See [Using the RMON and IEEE registers](#).

Address: 0h base + 2B0h offset = 2B0h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	RMON_R_P128TO255																															
W																																
Reset	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	

* Notes:

- x = Undefined at reset.

FEC_RMON_R_P128TO255 field descriptions

Field	Description
0–31 RMON_R_ P128TO255	RMON Rx 128 to 255 byte packets

46.5.67 RMON Rx 256 to 511 byte packets (FEC_RMONG_R_P256TO511)

See [Using the RMONG and IEEE registers](#).

Address: 0h base + 2B4h offset = 2B4h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	RMON_R_P256TO511																															
W																																
Reset	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	

* Notes:

- x = Undefined at reset.

FEC_RMONG_R_P256TO511 field descriptions

Field	Description
0–31 RMONG_R_ P256TO511	RMONG Rx 256 to 511 byte packets

46.5.68 RMONG Rx 512 to 1023 byte packets (FEC_RMONG_R_P512TO1023)

See [Using the RMONG and IEEE registers](#).

Address: 0h base + 2B8h offset = 2B8h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	RMON_R_P512TO1023																															
W																																
Reset	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	

* Notes:

- x = Undefined at reset.

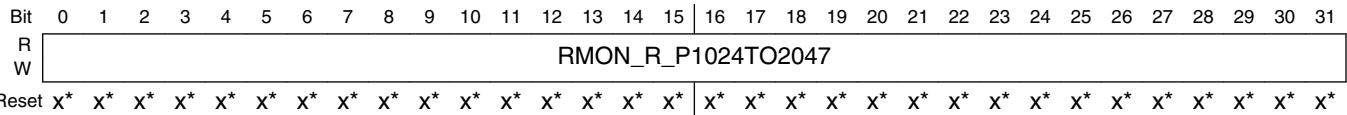
FEC_RMONG_R_P512TO1023 field descriptions

Field	Description
0–31 RMONG_R_ P512TO1023	RMONG Rx 512 to 1023 byte packets

46.5.69 RMON Rx 1024 to 2047 byte packets (FEC_RMON_R_P1024TO2047)

See [Using the RMON and IEEE registers](#).

Address: 0h base + 2BCh offset = 2BCh



- * Notes:
- x = Undefined at reset.

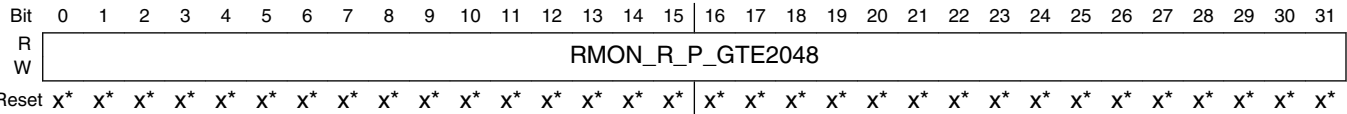
FEC_RMON_R_P1024TO2047 field descriptions

Field	Description
0–31 RMON_R_P1024TO2047	RMON Rx 1024 to 2047 byte packets

46.5.70 RMON Rx packets with > 2048 bytes (FEC_RMON_R_P_GTE2048)

See [Using the RMON and IEEE registers](#).

Address: 0h base + 2C0h offset = 2C0h



- * Notes:
- x = Undefined at reset.

FEC_RMON_R_P_GTE2048 field descriptions

Field	Description
0–31 RMON_R_P_GTE2048	RMON Rx packets with > 2048 bytes

46.5.71 RMON Rx octets (FEC_RMON_R_OCTETS)

See [Using the RMON and IEEE registers](#).

Address: 0h base + 2C4h offset = 2C4h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	RMON_R_OCTETS																															
W																																
Reset	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	

* Notes:

- x = Undefined at reset.

FEC_RMON_R_OCTETS field descriptions

Field	Description
0–31 RMON_R_OCTETS	RMON Rx octets

46.5.72 Count of received frames not counted correctly (FEC_IEEE_R_DROP)

See [Using the RMON and IEEE registers](#).

Address: 0h base + 2C8h offset = 2C8h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	IEEE_R_DROP																															
W																																
Reset	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	

* Notes:

- x = Undefined at reset.

FEC_IEEE_R_DROP field descriptions

Field	Description
0–31 IEEE_R_DROP	Count of received frames not counted correctly

46.5.73 Frames received OK (FEC_IEEE_R_FRAME_OK)

See [Using the RMON and IEEE registers](#).

Address: 0h base + 2CCh offset = 2CCh

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	IEEE_R_FRAME_OK																															
W																																
Reset	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	

* Notes:

- x = Undefined at reset.

FEC_IEEE_R_FRAME_OK field descriptions

Field	Description
0–31 IEEE_R_FRAME_OK	Frames received OK

46.5.74 Frames received with CRC error (FEC_IEEE_R_CRC)

See [Using the RMON and IEEE registers](#).

Address: 0h base + 2D0h offset = 2D0h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	IEEE_R_CRC																															
W																																
Reset	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	

* Notes:

- x = Undefined at reset.

FEC_IEEE_R_CRC field descriptions

Field	Description
0–31 IEEE_R_CRC	Frames received with CRC error

46.5.75 Frames received with alignment error (FEC_IEEE_R_ALIGN)

See [Using the RMON and IEEE registers](#).

Address: 0h base + 2D4h offset = 2D4h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	IEEE_R_ALIGN																															
W																																
Reset	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*

* Notes:

- x = Undefined at reset.

FEC_IEEE_R_ALIGN field descriptions

Field	Description
0–31 IEEE_R_ALIGN	Frames received with alignment error

46.5.76 Receive FIFO overflow count (FEC_IEEE_R_MACERR)

See [Using the RMON and IEEE registers](#).

Address: 0h base + 2D8h offset = 2D8h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	IEEE_R_MACERR																															
W																																
Reset	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*

* Notes:

- x = Undefined at reset.

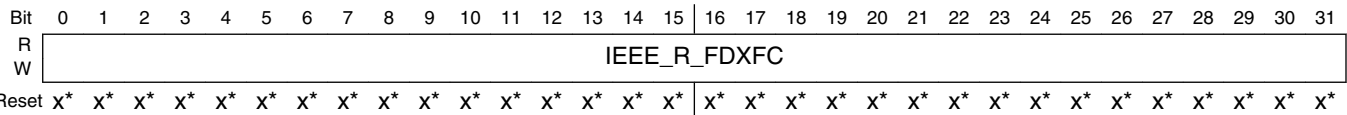
FEC_IEEE_R_MACERR field descriptions

Field	Description
0–31 IEEE_R_MACERR	Receive FIFO overflow count

46.5.77 Flow control pause frames received (FEC_IEEE_R_FDXFC)

See [Using the RMON and IEEE registers](#).

Address: 0h base + 2DCh offset = 2DCh



- * Notes:
- x = Undefined at reset.

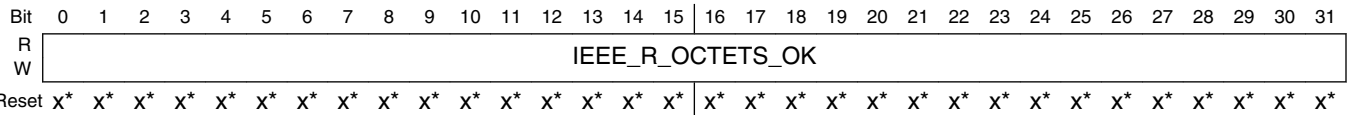
FEC_IEEE_R_FDXFC field descriptions

Field	Description
0–31 IEEE_R_FDXFC	Flow control pause frames received

46.5.78 Octet count for frames received without error (FEC_IEEE_R_OCTETS_OK)

See [Using the RMON and IEEE registers](#).

Address: 0h base + 2E0h offset = 2E0h



- * Notes:
- x = Undefined at reset.

FEC_IEEE_R_OCTETS_OK field descriptions

Field	Description
0–31 IEEE_R_OCTETS_OK	Octet count for frames received without error

46.5.79 Using the RMON and IEEE registers

In order to use the RMON and IEEE registers to monitor FEC performance, you must prepare them using the sequence below:

1. Write zeros to the registers.
2. Enable the FEC.

Until you complete this sequence, the registers are read/write, and their reset value is undefined. After you complete this sequence, the registers become read-only, and you may read them as necessary.

46.6 Functional description

This section describes the operation of the FEC, beginning with the description of two different types of MII frames, the buffer descriptors, the hardware and software initialization sequence, then the software (Ethernet driver) interface for transmitting and receiving frames.

Following the software initialization and operation sections are sections providing a detailed description of the functions of the FEC.

46.6.1 MII data frame

Ethernet/802.3 data frames transmitted across the MII have the format specified in [Figure 46-81](#) and [Table 46-85](#).

<inter-frame> <preamble> <sfd> <data> <efd>

Figure 46-81. MII data frame format

Table 46-85. MII data frame components

Component	Description
<inter-frame>	Inter-frame period. This is an unspecified amount of time during which no data activity occurs on the MII. The de-assertion of RX_DV and TX_EN indicate the absence of data activity.
<preamble>	Preamble that begins a frame. It has the following value: 10101010 10101010 10101010 10101010 10101010 10101010 10101010 The leftmost 1 represents the LSB of the byte.
<sfd>	Start of a frame. It has the value 10101011.
<data>	Data portion of the frame. It consists of N octets which corresponds to 2N nibbles being transmitted. The order of each nibble is defined in Figure 46-82 .

Table continues on the next page...

Table 46-85. MII data frame components (continued)

Component	Description
<efd>	End-of-frame delimiter. For TXD data, it is indicated by the de-assertion of the TX_EN signal. For RXD data, it is indicated by the de-assertion of the RX_DV signal.

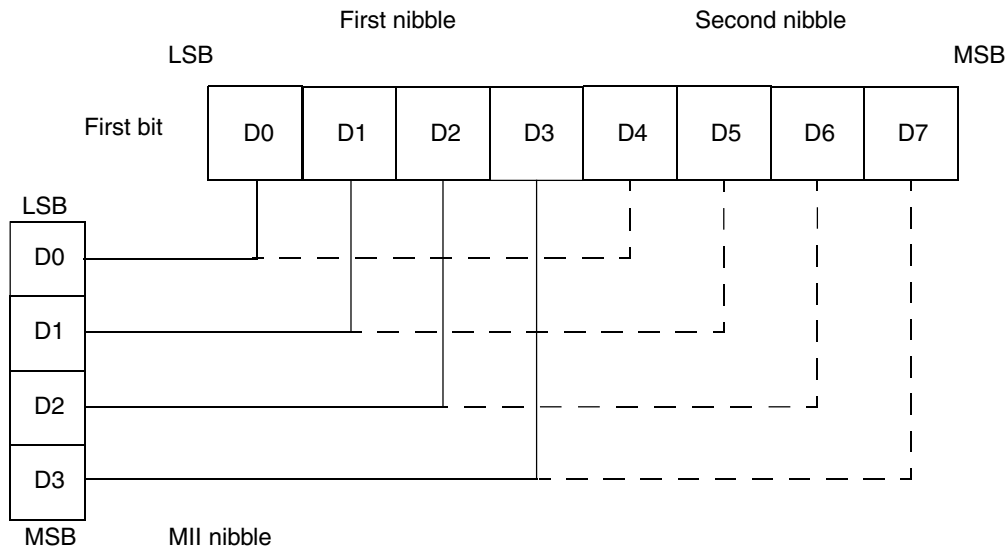


Figure 46-82. MII nibble/octet to octet/nibble mapping

46.6.2 MII management frame structure

A transceiver management frame being transmitted on the MII management interface uses the MDIO and MDC pins. A transaction or frame on this serial interface has the format specified in [Figure 46-83](#) and [Table 46-86](#).

<preamble> <st> <op> <phyad> <regad> <ta> <data> <idle>

Figure 46-83. MII management frame format

Table 46-86. MII management frame components

Component	Description
<preamble>	Optional preamble, consisting of a sequence of 32 continuous logic 1's.
<st>	Start of frame, consisting of a 01 pattern.
<op>	Operation code. 01 Write operation 10 Read instruction

Table continues on the next page...

Table 46-86. MII management frame components (continued)

Component	Description
<phyad>	Five-bit field that allows up to 32 PHYs to be addressed. The first address bit transmitted is the MSB of the address.
<regad>	Five-bit field that allows for 32 registers to be addressed within each PHY. The first register bit transmitted is the MSB of the address.
<ta>	Two-bit field that provides spacing between the register address field and the data field to avoid contention on the MDIO signal during a read operation.
<data>	16-bit data field. Bit 15 is the first data bit transmitted and received.
<idle>	Idle condition (MDIO is in the high-impedance state).

The MII management register set located in the PHY may consist of a basic register set and an extended register set as defined in [Table 46-87](#).

Table 46-87. MII management register set

Register address	Register name	Register set
0	Control	Basic
1	Status	
2-3	PHY identifier	Extended
4	Auto-negotiation advertisement	
5	AN link partner ability	
6	AN expansion	
7	AN next page transmit	
8-15	Reserved	
16-31	Vendor-specific	

46.6.3 Buffer descriptors

This section provides a description of the operation of the driver/DMA via the buffer descriptors. It is followed by a detailed description of the receive and transmit descriptor fields.

46.6.3.1 Driver/DMA operation with buffer descriptors

The data for the FEC frames resides in one or more memory buffers external to the FEC. Associated with each buffer is a buffer descriptor (BD), which contains a starting address (32-bit aligned pointer), data length, and status/control information (which contains the current state for the buffer). To permit maximum user flexibility, the BDs are also located in external memory and are read by the FEC DMA engine.

Software produces buffers by allocating/initializing memory and initializing buffer descriptors. Setting the RxBD[E] or TxBD[R] bit produces the buffer. Software writing to TDAR or RDAR tells the FEC that a buffer is placed in external memory for the transmit or receive data traffic, respectively. The hardware reads the BDs and consumes the buffers after they have been produced. After the data DMA is complete and the DMA engine writes the buffer descriptor status bits, hardware clears RxBD[E] or TxBD[R] to signal the buffer has been consumed. Software may poll the BDs to detect when the buffers are consumed or may rely on the buffer/frame interrupts. The driver may process these buffers, and they can return to the free list.

The ECR[ETHER_EN] bit operates as a reset to the BD/DMA logic. When ECR[ETHER_EN] is cleared, the DMA engine BD pointers are reset to point to the starting transmit and receive BDs. The buffer descriptors are not initialized by hardware during reset. At least one transmit and receive buffer descriptor must be initialized by software before ECR[ETHER_EN] is set.

The buffer descriptors operate as two separate rings. ERDSR defines the starting address for receive BDs and ETDSR defines the starting address for transmit BDs. The wrap (W) bit defines the last buffer descriptor in each ring. When W is set, the next descriptor in the ring is at the location pointed to by ERDSR and ETDSR for the receive and transmit rings, respectively. Buffer descriptor rings must start on a 32-bit boundary; however, it is recommended they be 128-bit aligned.

46.6.3.1.1 Driver/DMA operation with transmit BDs

Typically, a transmit frame is divided between multiple buffers. An example is to have an application payload in one buffer, TCP header in a second buffer, IP header in a third buffer, and Ethernet/IEEE 802.3 header in a fourth buffer. The Ethernet MAC does not prepend the Ethernet header (destination address, source address, length/type field(s)), so the driver must provide this in one of the transmit buffers. The Ethernet MAC can append the Ethernet CRC to the frame. TxBD[TC], which must be set by the driver, determines whether the MAC or driver appends the CRC.

The driver (TxBD software producer) should set up Tx BDs so a complete transmit frame is given to the hardware at once. If a transmit frame consists of three buffers, the BDs should be initialized with pointer, length, and control (W, L, TC, ABC) and then the TxBD[R] bit should be set in reverse order (third, second, then first BD) to ensure that the complete frame is ready in memory before the DMA begins. If the TxBDs are set up in order, the DMA controller could DMA the first BD before the second was made available, potentially causing a transmit FIFO underrun.

In the FEC, the driver notifies the DMA that new transmit frame(s) are available by writing to TDAR. When this register is written to (data value is not significant), the FEC RISC tells the DMA to read the next transmit BD in the ring. After started, the RISC + DMA continues to read and interpret transmit BDs in order and DMA the associated buffers until a transmit BD is encountered with the R bit cleared. At this point, the FEC polls this BD one more time. If the R bit is cleared the second time, RISC stops the transmit descriptor read process until software sets up another transmit frame and writes to TDAR.

When the DMA of each transmit buffer is complete, the DMA writes back to the BD to clear the R bit, indicating that the hardware consumer is finished with the buffer.

46.6.3.1.2 Driver/DMA operation with receive BDs

Unlike transmit, the length of the receive frame is unknown by the driver ahead of time. Therefore, the driver must set a variable to define the length of all receive buffers. In the FEC, this variable is written to the EMRBR register.

The driver (RxBD software producer) should set up some number of empty buffers for the Ethernet by initializing the address field and the E and W bits of the associated receive BDs. The hardware (receive DMA) consumes these buffers by filling them with data as frames are received and clearing the E bit and writing to the L bit (1 indicates last buffer in frame), the frame status bits (if L is set), and the length field.

If a receive frame spans multiple receive buffers, the L bit is only set for the last buffer in the frame. For non-last buffers, the length field in the receive BD is written by the DMA (at the same time the E bit is cleared) with the default receive buffer length value. For end-of-frame buffers, the receive BD is written with L set and information written to the status bits (M, BC, MC, LG, NO, CR, OV, TR). Some of the status bits are error indicators which, if set, indicate the receive frame should be discarded and not given to higher layers. The frame status/length information is written into the receive FIFO following the end of the frame (as a single 32-bit word) by the receive logic. The length field for the end of frame buffer is written with the length of the entire frame, not only the length of the last buffer.

For simplicity, the driver may assign a large enough default receive buffer length to contain an entire frame, keeping in mind that a malfunction on the network or out-of-spec implementation could result in giant frames. Frames of 2K (2048) bytes or larger are truncated by the FEC at 2047 bytes so software never sees a receive frame larger than 2047 bytes.

Similar to transmit, the FEC polls the receive descriptor ring after the driver sets up receive BDs and writes to the RDAR register. As frames are received, the FEC fills receive buffers and updates the associated BDs, then reads the next BD in the receive

descriptor ring. If the FEC reads a receive BD and finds the E bit cleared, it polls this BD once more. If RxBDE is clear a second time, FEC stops reading receive BDs until the driver writes to RDAR.

46.6.3.2 Ethernet Receive Buffer Descriptor (RxBDE)

In the RxBDE, the user initializes the E and W bits in the first longword and the pointer in the second longword. When the buffer has been DMA'd, the Ethernet controller modifies the E, L, M, BC, MC, LG, NO, CR, OV, and TR bits and writes the length of the used portion of the buffer in the first longword. The M, BC, MC, LG, NO, CR, OV, and TR bits in the first longword of the buffer descriptor are only modified by the Ethernet controller when the L bit is set.

Table 46-88. Receive Buffer Descriptor (RxBDE)

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Offset+0	E	RO1	W	RO2	L	—	—	M	BC	MC	LG	NO	—	CR	OV	TR
Offset+2	Data Length															
Offset+4	Rx Data Buffer Pointer - A[31:16]															
Offset+6	Rx Data Buffer Pointer - A[15:0]															

Table 46-89. Receive Buffer Descriptor field descriptions

Word	Field	Description
Offset + 0	E	Empty. Written by the FEC (=0) and user (=1). 0: The data buffer associated with this BD is filled with received data, or data reception has aborted due to an error condition. The status and length fields have been updated as required. 1: The data buffer associated with this BD is empty, or reception is currently in progress.
Offset + 0	RO1	Receive software ownership. This field is reserved for use by software. This read/write bit is not modified by hardware, nor does its value affect hardware.
Offset + 0	W	Wrap. Written by user. 0: The next buffer descriptor is found in the consecutive location 1: The next buffer descriptor is found at the location defined in ERDSR.
Offset + 0	RO2	Receive software ownership. This field is reserved for use by software. This read/write bit is not modified by hardware, nor does its value affect hardware.
Offset + 0	L	Last in frame. Written by the FEC. 0: The buffer is not the last in a frame. 1: The buffer is the last in a frame.

Table continues on the next page...

Table 46-89. Receive Buffer Descriptor field descriptions (continued)

Word	Field	Description
Offset + 0	M	Miss. Written by the FEC. This bit is set by the FEC for frames accepted in promiscuous mode, but flagged as a miss by the internal address recognition. Therefore, while in promiscuous mode, you can use the M-bit to quickly determine whether the frame was destined to this station. This bit is valid only if the L-bit is set and the PROM bit is set. 0: The frame was received because of an address recognition hit. 1: The frame was received because of promiscuous mode.
Offset + 0	BC	Set if the DA is broadcast (FFFF_FFFF_FFFF).
Offset + 0	MC	Set if the DA is multicast and not BC.
Offset + 0	LG	Rx frame length violation. Written by the FEC. A frame length greater than RCR[MAX_FL] was recognized. This bit is valid only if the L-bit is set. The receive data is not altered in any way unless the length exceeds 2047 bytes.
Offset + 0	NO	Receive non-octet aligned frame. Written by the FEC. A frame that contained a number of bits not divisible by 8 was received, and the CRC check that occurred at the preceding byte boundary generated an error. This bit is valid only if the L-bit is set. If this bit is set, the CR bit is not set.
Offset + 0	CR	Receive CRC error. Written by the FEC. This frame contains a CRC error and is an integral number of octets in length. This bit is valid only if the L-bit is set.
Offset + 0	OV	Overflow. Written by the FEC. A receive FIFO overflow occurred during frame reception. If this bit is set, the other status bits, M, LG, NO, CR, and CL lose their normal meaning and are zero. This bit is valid only if the L-bit is set.
Offset + 0	TR	Set if the receive frame is truncated (frame length > 2047 bytes). If the TR bit is set, the frame must be discarded and the other error bits must be ignored as they may be incorrect.
Offset + 2	Data Length	Data length. Written by the FEC. Data length is the number of octets written by the FEC into this BD's data buffer if L equals 0 (the value is equal to EMRBR), or the length of the frame including CRC if L is set. It is written by the FEC once as the BD is closed.
Offset + 4	A[31:16]	RX data buffer pointer, bits [31:16] ¹
Offset + 6	A[15:0]	RX data buffer pointer, bits [15:0]

1. The receive buffer pointer, containing the address of the associated data buffer, must always be evenly divisible by 16. The buffer must reside in memory external to the FEC. The Ethernet controller never modifies this value.

Note

When the software driver sets an E bit in one or more receive descriptors, the driver should follow with a write to RDAR.

46.6.3.3 Ethernet Transmit Buffer Descriptor (TxBD)

Data is presented to the FEC for transmission by arranging it in buffers referenced by the channel's TxBDs. The Ethernet controller confirms transmission by clearing the ready bit (TxBD[R]) when DMA of the buffer is complete. In the TxBD, the user initializes the R, W, L, and TC bits and the length (in bytes) in the first longword and the buffer pointer in the second longword.

Functional description

The FEC clears the R bit when the buffer is transferred. Status bits for the buffer/frame are not included in the transmit buffer descriptors. Transmit frame status is indicated via individual interrupt bits (error conditions) and in statistic counters in the MIB block.

Table 46-90. Transmit Buffer Descriptor (TxBD)

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Offset+0	R	TO1	W	TO2	L	TC	ABC	—	—	—	—	—	—	—	—	—
Offset+2	Data Length															
Offset+4	Tx Data Buffer Pointer - A[31:16]															
Offset+6	Tx Data Buffer Pointer - A[15:0]															

Table 46-91. Transmit Buffer Descriptor field descriptions

Word	Field	Description
Offset + 0	R	Ready. Written by the FEC and you. 0: The data buffer associated with this BD is not ready for transmission. You are free to manipulate this BD or its associated data buffer. The FEC clears this bit after the buffer has been transmitted or after an error condition is encountered. 1: The data buffer, prepared for transmission by you, has not been transmitted or currently transmits. You may write no fields of this BD after this bit is set.
Offset + 0	TO1	Transmit software ownership. This field is reserved for software use. This read/write bit is not modified by hardware nor does its value affect hardware.
Offset + 0	W	Wrap. Written by user. 0: The next buffer descriptor is found in the consecutive location 1: The next buffer descriptor is found at the location defined in ETDSR.
Offset + 0	TO2	Transmit software ownership. This field is reserved for use by software. This read/write bit is not modified by hardware nor does its value affect hardware.
Offset + 0	L	Last in frame. Written by user. 0: The buffer is not the last in the transmit frame 1: The buffer is the last in the transmit frame
Offset + 0	TC	Transmit CRC. Written by user (only valid if L is set). 0: End transmission immediately after the last data byte 1: Transmit the CRC sequence after the last data byte
Offset + 0	ABC	Append bad CRC. Written by user (only valid if L is set). 0: No effect 1: Transmit the CRC sequence inverted after the last data byte (regardless of TC value)
Offset + 2	Data Length	Data length, written by user. Data length is the number of octets the FEC should transmit from this BD's data buffer. It is never modified by the FEC.
Offset + 4	A[31:16]	Tx data buffer pointer, bits [31:16] 1
Offset + 6	A[15:0]	Tx data buffer pointer, bits [15:0]

1. The transmit buffer pointer, containing the address of the associated data buffer, must always be evenly divisible by 4. The buffer must reside in memory external to the FEC. This value is never modified by the Ethernet controller.

Note

After the software driver has set up the buffers for a frame, it should set up the corresponding BDs. The last step in setting up the BDs for a transmit frame is setting the R bit in the first BD for the frame. The driver must follow that with a write to TDAR that triggers the FEC to poll the next BD in the ring.

46.6.4 Initialization sequence

This section describes which registers are reset due to hardware reset, which registers are reset by the FEC RISC, and what locations you must initialize prior to enabling the FEC.

46.6.4.1 Hardware Controlled Initialization

In the FEC, hardware resets registers and control logic that generate interrupts. A hardware reset negates output signals and resets general configuration bits.

Other registers reset when the ECR[ETHER_EN] bit is cleared (which is accomplished by a hard reset or software to halt operation). By clearing ECR[ETHER_EN], configuration control registers such as the TCR and RCR are not reset, but the entire data path is reset.

Table 46-92. ECR[ETHER_EN] De-Assertion Effect on FEC

Register/Machine	Reset Value
XMIT block	Transmission is aborted (bad CRC appended)
RECV block	Receive activity is aborted
DMA block	All DMA activity is terminated
RDAR	Cleared
TDAR	Cleared
Descriptor Controller block	Halt operation

46.6.5 User initialization (Prior to Setting ECR[ETHER_EN])

You need to initialize portions the FEC prior to setting the ECR[ETHER_EN] bit. The exact values depend on the particular application. The sequence is not important.

The following registers require initialization:

- EIMR
- EIR (clear by writing FFFF_FFFFh)
- TFWR (optional)
- IALR / IAUR
- GAUR / GALR
- PALR / PAUR (only needed for full duplex flow control)
- OPD (only needed for full duplex flow control)
- RCR
- TCR
- MSCR (optional)
- MIB_RAM (clear)
- FRSR (optional)
- EMRBR
- ERDSR
- ETDSR
- Transmit Descriptor ring (empty)
- Receive Descriptor ring (empty)

46.6.6 Microcontroller initialization

In the FEC, the descriptor control RISC initializes some registers after ECR[ETHER_EN] is asserted. After the microcontroller initialization sequence is complete, hardware is ready for operation.

The sequence is as follows:

1. Initialize the backoff random number seed.
2. Activate receiver.
3. Activate transmitter.

4. Clear transmit FIFO.
5. Clear receive FIFO.
6. Initialize the transmit ring pointer.
7. Initialize the receive ring pointer.
8. Initialize FIFO count registers.

46.6.7 User initialization (after setting ECR[ETHER_EN])

After setting ECR[ETHER_EN], you can set up the buffer/frame descriptors and write to TDAR and RDAR. See [Buffer descriptors](#), for more details.

46.6.8 Network interface options

The FEC supports an MII and reduced MII interface for 10/100 Mbps Ethernet, as well as a 7-wire serial interface for 10 Mbps Ethernet. The RCR[MII_MODE] and RCR[RMII_MODE] bits select the interface mode. In MII mode (RCR[MII_MODE]=1 and RCR[RMII_MODE]=0), there are 18 signals defined by the IEEE 802.3 standard and supported by the EMAC. The following table shows these signals.

Table 46-93. MII mode

Signal description	EMAC pin
Transmit Clock	FEC_TXCLK
Transmit Enable	FEC_TXEN
Transmit Data	FEC_TXD[3:0]
Transmit Error	FEC_TXER
Collision	FEC_COL
Carrier Sense	FEC_CRS
Receive Clock	FEC_RXCLK
Receive Data Valid	FEC_RXDV
Receive Data	FEC_RXD[3:0]
Receive Error	FEC_RXER
Management Data Clock	FEC_MDC
Management Data Input/Output	FEC_MDIO

In RMII mode (RCR[MII_MODE]=1 and RCR[RMII_MODE]=1), the EMAC supports 8 external signals. These signals are shown in the table below.

Table 46-94. RMII mode configuration

Signal description	EMAC pin
Reference Clock	FEC_TXCLK
Transmit Enable	FEC_TXEN
Transmit Data	FEC_TXD[1:0]
Receive Data Valid	FEC_RXDV
Receive Data	FEC_RXD[1:0]
Receive Error	FEC_RXER

The 7-wire serial mode interface (RCR[MII_MODE]=1 and RCR[RMII_MODE]=0 or 1) is generally referred to as AMD mode. The following table shows the 7-wire mode connections to the external transceiver.

Table 46-95. 7-Wire mode configuration

Signal description	EMAC pin
Transmit Clock	FEC_TXCLK
Transmit Enable	FEC_TXEN
Transmit Data	FEC_TXD[0]
Collision	FEC_COL
Receive Clock	FEC_RXCLK
Receive Data Valid	FEC_RXDV
Receive Data	FEC_RXD[0]

46.6.9 FEC frame transmission

The Ethernet transmitter is designed to work with almost no intervention from software. After ECR[ETHER_EN] is set and data appears in the transmit FIFO, the Ethernet MAC can transmit onto the network. The Ethernet controller transmits bytes least significant bit (lsb) first.

When the transmit FIFO fills to the watermark (defined by TFWR), MAC transmit logic asserts FEC_TXEN and starts transmitting the (PA) sequence, the start frame delimiter (SFD), and then the frame information from the FIFO. However, the controller defers the transmission if the network is busy (FEC_CRS is asserted). Before transmitting, the controller waits for carrier sense to become inactive, then determines if carrier sense stays inactive for 60 bit times. If so, transmission begins after waiting an additional 36 bit times (96 bit times after carrier sense originally became inactive). See "Transmission errors" for more details.

If a collision occurs during transmission of the frame (half duplex mode), the Ethernet controller follows the specified backoff procedures and attempts to retransmit the frame until the retry limit is reached. The transmit FIFO stores at least the first 64 bytes of the transmit frame, so they do not have to be retrieved from system memory in case of a collision. This improves bus utilization and latency in case immediate retransmission is necessary.

When all the frame data is transmitted, FCS (frame check sequence) or 32-bit cyclic redundancy check (CRC) bytes are appended if the TC bit is set in the transmit frame control word. If the ABC bit is set in the transmit frame control word, a bad CRC is appended to the frame data regardless of the TC bit value. Following the transmission of the CRC, the Ethernet controller writes the frame status information to the MIB block. Transmit logic automatically pads short frames (if the TC bit in the transmit buffer descriptor for the end of frame buffer is set).

Settings in the EIMR determine interrupts generated to the buffer (TXB) and frame (TXF).

The transmit error interrupts are HBERR, BABT, LC, RL, and UN. If the transmit frame length exceeds MAX_FL bytes, BABT interrupt is asserted. However, the entire frame is transmitted (no truncation).

To pause transmission, set TCR[GTS] (graceful transmit stop). The FEC transmitter stops immediately if transmission is not in progress; otherwise, it continues transmission until the current frame finishes or terminates with a collision. After the transmitter has stopped, the GRA (graceful stop complete) interrupt is asserted. If TCR[GTS] is cleared, the FEC resumes transmission with the next frame.

46.6.9.1 Duplicate frame transmission

The FEC fetches transmit buffer descriptors (TxBDs) and the corresponding transmit data continuously until the transmit FIFO is full. It does not determine whether the TxBD to be fetched is already being processed internally (as a result of a wrap). As the FEC nears the end of the transmission of one frame, it begins to DMA the data for the next frame. To remain one BD ahead of the DMA, it also fetches the TxBD for the next frame. It is possible that the FEC fetches from memory a BD that has already been processed but not yet written back (it is read a second time with the R bit remains set). In this case, the data is fetched and transmitted again.

Using at least three TxBDs fixes this problem for large frames, but not for small frames. To ensure correct operation for large or small frames, one of the following must be true:

- The FEC software driver ensures that there is always at least one TxBD with the ready bit cleared.
- Every frame uses more than one TxBD and every TxBD but the last is written back immediately after the data is fetched.
- The FEC software driver ensures a minimum frame size, n . The minimum number of TxBDs is then $(\text{Tx FIFO Size} \div (n+4))$ rounded up to the nearest integer (though the result cannot be less than three). The default Tx FIFO size is 192 bytes; this size is programmable.

46.6.10 FEC frame reception

The FEC receiver works with almost no intervention from the host and can perform address recognition, CRC checking, short frame checking, and maximum frame length checking. The Ethernet controller receives serial data lsb first.

When the driver enables the FEC receiver by setting ECR[ETHER_EN], it immediately starts processing receive frames. When FEC_RXDV is asserted, the receiver first checks for a valid PA/SFD header. If the PA/SFD is valid, it is stripped and the receiver processes the frame. If a valid PA/SFD is not found, the frame is ignored.

In serial mode, the first 16 bit times of RX_D0 following assertion of FEC_RXDV are ignored. Following the first 16 bit times, the data sequence is checked for alternating 1/0s. If a 11 or 00 data sequence is detected during bit times 17 to 21, the remainder of the frame is ignored. After bit time 21, the data sequence is monitored for a valid SFD (11). If a 00 is detected, the frame is rejected. When a 11 is detected, the PA/SFD sequence is complete.

In MII mode, the receiver checks for at least one byte matching the SFD. Zero or more PA bytes may occur, but if a 00 bit sequence is detected prior to the SFD byte, the frame is ignored.

After the first 6 bytes of the frame are received, the FEC performs address recognition on the frame.

After a collision window (64 bytes) of data is received and if address recognition has not rejected the frame, the receive FIFO signals the frame is accepted and may be passed on to the DMA. If the frame is a runt (due to collision) or is rejected by address recognition, the receive FIFO is notified to reject the frame. Therefore, no collision fragments are presented to you except late collisions, which indicate serious LAN problems.

During reception, the Ethernet controller checks for various error conditions and after the entire frame is written into the FIFO, a 32-bit frame status word is written into the FIFO. This status word contains the M, BC, MC, LG, NO, CR, OV, and TR status bits, and the frame length. See "Reception errors" for more details.

Receive buffer (RXB) and frame interrupts (RFINT) may be generated if enabled by the EIMR register. A receive error interrupt is a babbling receiver error (BABR). Receive frames are not truncated if they exceed the max frame length (MAX_FL); however, the BABR interrupt occurs and the LG bit in the receive buffer descriptor (RxBD) is set. See [Ethernet Receive Buffer Descriptor \(RxBD\)](#), for more details.

When the receive frame is complete, the FEC sets the L-bit in the RxBD, writes the other frame status bits into the RxBD, and clears the E-bit. The Ethernet controller next generates a maskable interrupt (RFINT bit in EIR, maskable by RFIEN bit in EIMR), indicating that a frame is received and is in memory. The Ethernet controller then waits for a new frame.

46.6.11 Ethernet address recognition

The FEC filters the received frames based on destination address (DA) type — individual (unicast), group (multicast), or broadcast (all-ones group address). The difference between an individual address and a group address is determined by the I/G bit in the destination address field. A flowchart for address recognition on received frames appears in the figures below.

Address recognition is accomplished through the use of the receive block and microcode running on the microcontroller. The flowchart shown in [Figure 46-84](#) illustrates the address recognition decisions made by the receive block, while [Figure 46-85](#) illustrates the decisions made by the microcontroller.

If the DA is a broadcast address and broadcast reject (RCR[BC_REJ]) is cleared, then the frame is accepted unconditionally, as shown in [Figure 46-84](#). Otherwise, if the DA is not a broadcast address, then the microcontroller runs the address recognition subroutine, as shown in [Figure 46-85](#).

If the DA is a group (multicast) address and flow control is disabled, then the microcontroller performs a group hash table lookup using the 64-entry hash table programmed in GAUR and GALR. If a hash match occurs, the receiver accepts the frame.

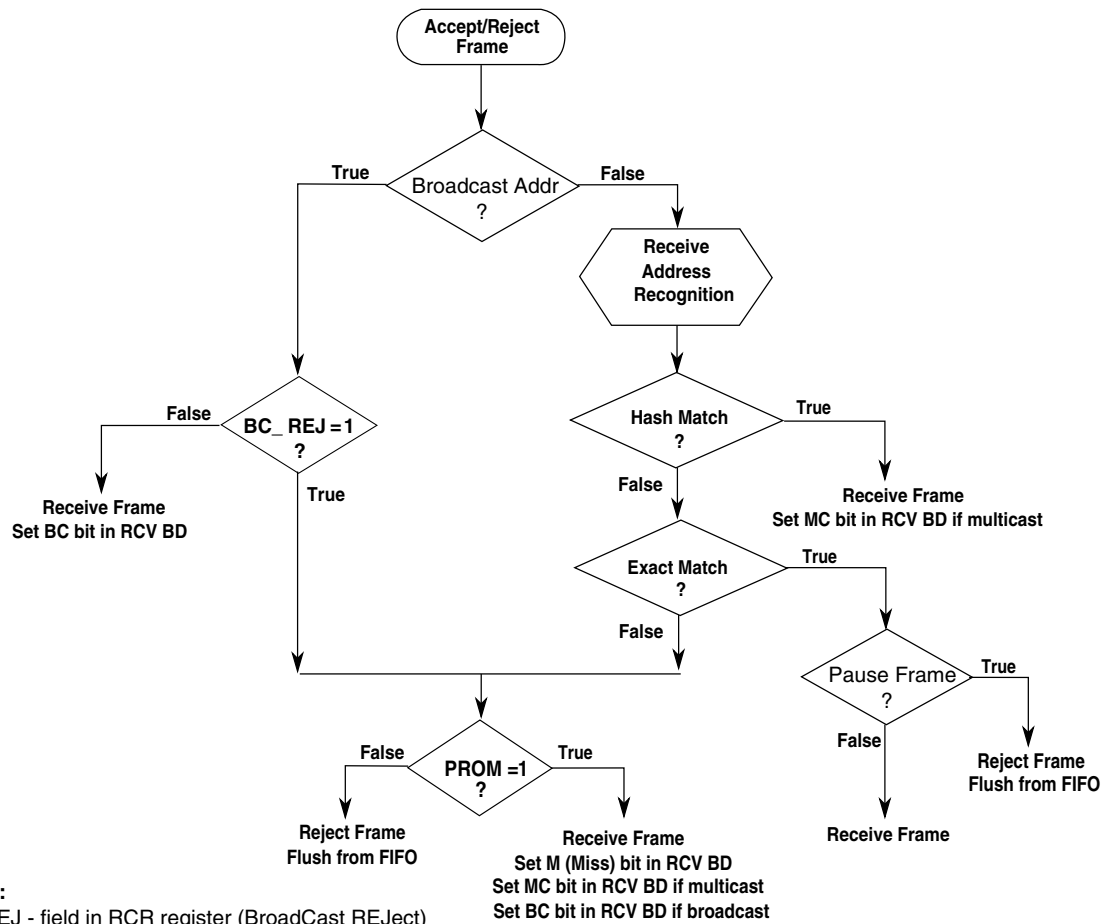
If flow control is enabled, the microcontroller does an exact address match check between the DA and the designated PAUSE DA (01:80:C2:00:00:01). If the receive block determines the received frame is a valid PAUSE frame, the frame is rejected. The receiver detects a PAUSE frame with the DA field set to the designated PAUSE DA or the unicast physical address.

If the DA is the individual (unicast) address, the microcontroller performs an individual exact match comparison between the DA and 48-bit physical address that you program in the PALR and PAUR registers. If an exact match occurs, the frame is accepted; otherwise, the microcontroller does an individual hash table lookup using the 64-entry hash table programmed in registers, IAUR and IALR. In the case of an individual hash match, the frame is accepted. Again, the receiver accepts or rejects the frame based on PAUSE frame detection, shown in [Figure 46-84](#).

If neither a hash match (group or individual) nor an exact match (group or individual) occur, and if promiscuous mode is enabled (RCR[PROM] set), the frame is accepted and the MISS bit in the receive buffer descriptor is set; otherwise, the frame is rejected.

Similarly, if the DA is a broadcast address, broadcast reject (RCR[BC_REJ]) is asserted, and promiscuous mode is enabled, the frame is accepted and the MISS bit in the receive buffer descriptor is set; otherwise, the frame is rejected.

In general, when a frame is rejected, it is flushed from the FIFO.

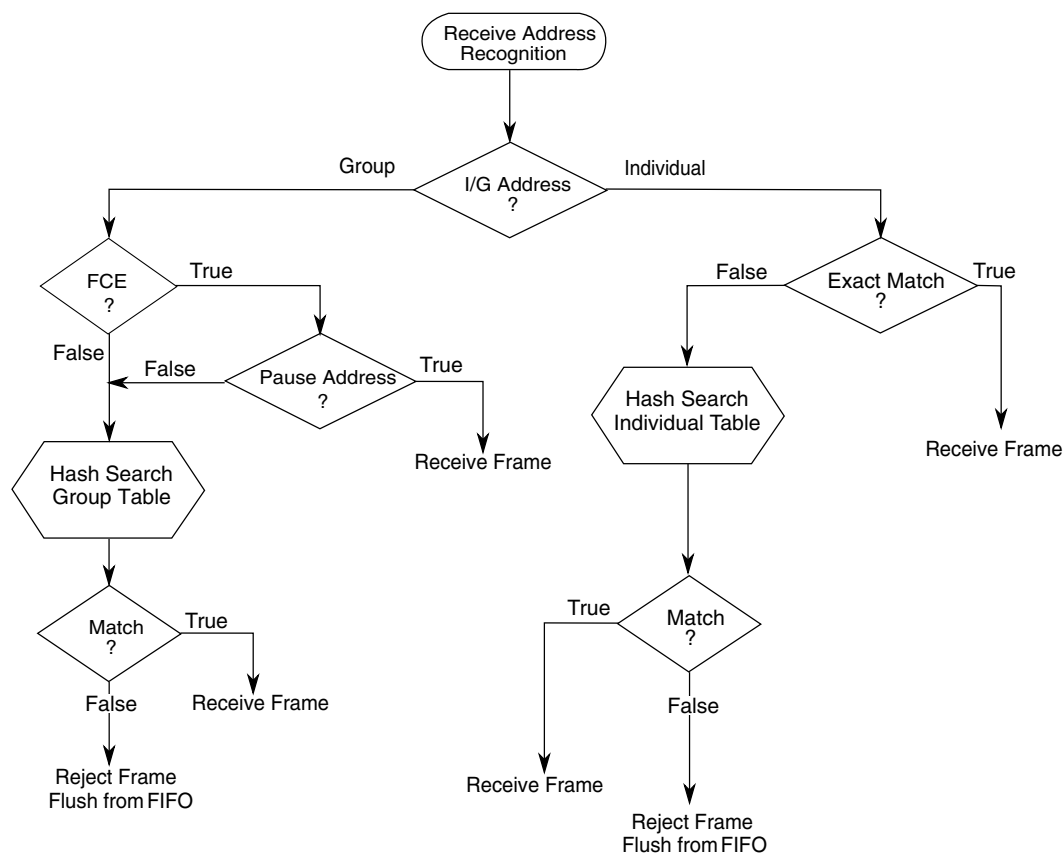
**Notes:**

BC_REJ - field in RCR register (BroadCast REJect)

PROM - field in RCR register (PROMiscuous mode)

Pause Frame - valid PAUSE frame received

Figure 46-84. Ethernet address recognition—receive block decisions

**Notes:**

FCE - field in RCR register (flow control enable)

I/G - Individual/Group bit in destination address (lsb in first byte received in MAC frame)

Figure 46-85. Ethernet address recognition—microcode decisions

46.6.12 Hash algorithm

The hash table algorithm used in the group and individual hash filtering operates as follows. The 48-bit destination address is mapped into one of 64 bits, represented by 64 bits stored in GAUR, GALR (group address hash match), or IAUR, IALR (individual address hash match). This mapping is performed by passing the 48-bit address through the on-chip 32-bit CRC generator and selecting the six most significant bits of the CRC-encoded result to generate a number between 0 and 63. The msb of the CRC result selects GAUR (msb = 1) or GALR (msb = 0). The five least significant bits of the hash result select the bit within the selected register. If the CRC generator selects a bit set in the hash table, the frame is accepted; otherwise, it is rejected.

For example, if eight group addresses are stored in the hash table and random group addresses are received, the hash table prevents roughly 56/64 (87.5%) of the group address frames from reaching memory. Those that do reach memory must be further filtered by the processor to determine if they truly contain one of the eight desired addresses.

The effectiveness of the hash table declines as the number of addresses increases.

The user must initialize the hash table registers. Use this CRC32 polynomial to compute the hash:

$$X^{32} + X^{26} + X^{23} + X^{22} + X^{16} + X^{12} + X^{11} + X^{10} + X^8 + X^7 + X^5 + X^4 + X^2 + X + 1$$

Figure 46-86. CRC32 polynomial for hash computation

The following table contains example destination addresses and corresponding hash values.

Table 46-96. Destination address to 6-bit hash

48-BIT DA	6-bit hash (in hex)	Hash decimal value
65FF_FFFF_FFFF	0h	0
55FF_FFFF_FFFF	1h	1
15FF_FFFF_FFFF	2h	2
35FF_FFFF_FFFF	3h	3
B5FF_FFFF_FFFF	4h	4
95FF_FFFF_FFFF	5h	5
D5FF_FFFF_FFFF	6h	6
F5FF_FFFF_FFFF	7h	7
DBFF_FFFF_FFFF	8h	8
FBFF_FFFF_FFFF	9h	9
BBFF_FFFF_FFFF	Ah	10
8BFF_FFFF_FFFF	Bh	11
0BFF_FFFF_FFFF	Ch	12
3BFF_FFFF_FFFF	Dh	13
7BFF_FFFF_FFFF	Eh	14
5BFF_FFFF_FFFF	Fh	15
27FF_FFFF_FFFF	10h	16
07FF_FFFF_FFFF	11h	17
57FF_FFFF_FFFF	12h	18
77FF_FFFF_FFFF	13h	19
F7FF_FFFF_FFFF	14h	20
C7FF_FFFF_FFFF	15h	21
97FF_FFFF_FFFF	16h	22
A7FF_FFFF_FFFF	17h	23
99FF_FFFF_FFFF	18h	24
B9FF_FFFF_FFFF	19h	25
F9FF_FFFF_FFFF	1Ah	26
C9FF_FFFF_FFFF	1Bh	27

Table continues on the next page...

Table 46-96. Destination address to 6-bit hash (continued)

48-BIT DA	6-bit hash (in hex)	Hash decimal value
59FF_FFFF_FFFF	1Ch	28
79FF_FFFF_FFFF	1Dh	29
29FF_FFFF_FFFF	1Eh	30
19FF_FFFF_FFFF	1Fh	31
D1FF_FFFF_FFFF	20h	32
F1FF_FFFF_FFFF	21h	33
B1FF_FFFF_FFFF	22h	34
91FF_FFFF_FFFF	23h	35
11FF_FFFF_FFFF	24h	36
31FF_FFFF_FFFF	25h	37
71FF_FFFF_FFFF	26h	38
51FF_FFFF_FFFF	27h	39
7FFF_FFFF_FFFF	28h	40
4FFF_FFFF_FFFF	29h	41
1FFF_FFFF_FFFF	2Ah	42
3FFF_FFFF_FFFF	2Bh	43
BFFF_FFFF_FFFF	2Ch	44
9FFF_FFFF_FFFF	2Dh	45
DFFF_FFFF_FFFF	2Eh	46
EFFF_FFFF_FFFF	2Fh	47
93FF_FFFF_FFFF	30h	48
B3FF_FFFF_FFFF	31h	49
F3FF_FFFF_FFFF	32h	50
D3FF_FFFF_FFFF	33h	51
53FF_FFFF_FFFF	34h	52
73FF_FFFF_FFFF	35h	53
23FF_FFFF_FFFF	36h	54
13FF_FFFF_FFFF	37h	55
3DFF_FFFF_FFFF	38h	56
0DFF_FFFF_FFFF	39h	57
5DFF_FFFF_FFFF	3Ah	58
7DFF_FFFF_FFFF	3Bh	59
FDFF_FFFF_FFFF	3Ch	60
DDFF_FFFF_FFFF	3Dh	61
9DFF_FFFF_FFFF	3Eh	62
BDFF_FFFF_FFFF	3Fh	63

46.6.13 Full duplex flow control

Full-duplex flow control allows you to transmit pause frames and to detect received pause frames. Upon detection of a pause frame, MAC data frame transmission stops for a given pause duration.

To enable PAUSE frame detection, the FEC must operate in full-duplex mode (TCR[FDEN] set) with flow control (RCR[FCE] set). The FEC detects a pause frame when the fields of the incoming frame match the pause frame specifications, as shown in the following table. In addition, the receive status associated with the frame should indicate that the frame is valid.

Table 46-97. PAUSE frame field specification

48-bit destination address	0180_C200_0001h or Physical Address
48-bit source address	Any
16-bit type	8808h
16-bit opcode	0001h
16-bit PAUSE duration	0000h — FFFFh

The receiver and microcontroller modules perform PAUSE frame detection. The microcontroller runs an address recognition subroutine to detect the specified pause frame destination address, while the receiver detects the type and opcode pause frame fields. On detection of a pause frame, TCR[GTS] is set by the FEC internally. When transmission has paused, the EIR[GRA] interrupt is asserted and the pause timer begins to increment. The pause timer uses the transmit backoff timer hardware for tracking the appropriate collision backoff time in half-duplex mode. The pause timer increments once every slot time, until OPD[PAUSE_DUR] slot times have expired. On OPD[PAUSE_DUR] expiration, TCR[GTS] is cleared allowing MAC data frame transmission to resume. The receive flow control pause status bit (TCR[RFC_PAUSE]) is set while the transmitter pauses due to reception of a pause frame.

To transmit a pause frame, the FEC must operate in full-duplex mode and you must set flow control pause (TCR[TFC_PAUSE]). After TCR[TFC_PAUSE] is set, the transmitter sets TCR[GTS] internally. When the transmission of data frames stops, the EIR[GRA] (graceful stop complete) interrupt asserts and the pause frame is transmitted. TCR[TFC_PAUSE,GTS] are then cleared internally.

You must specify the desired pause duration in the OPD register.

When the transmitter pauses due to receiver/microcontroller pause frame detection, TCR[TFC_PAUSE] may remain set and cause the transmission of a single pause frame. In this case, the EIR[GRA] interrupt is not asserted.

46.6.14 Inter-Packet Gap (IPG) time

The minimum inter-packet gap time for back-to-back transmission is 96 bit times. After completing a transmission or after the backoff algorithm completes, the transmitter waits for carrier sense to be negated before starting its 96 bit time IPG counter. Frame transmission may begin 96 bit times after carrier sense is negated if it stays negated for at least 60 bit times. If carrier sense asserts during the last 36 bit times, it is ignored and a collision occurs.

The receiver accepts back-to-back frames with a minimum spacing of at least 28 bit times. If an inter-packet gap between receive frames is less than 28 bit times, the receiver may discard the following frame.

46.6.15 Collision managing

If a collision occurs during frame transmission, the Ethernet controller continues the transmission for at least 32 bit times, transmitting a JAM pattern consisting of 32 ones. If the collision occurs during the sequence, a JAM pattern is sent after the end of the preamble sequence.

If a collision occurs within 512 bit times (one slot time), the retry process is initiated. The transmitter waits a random number of slot times. If a collision occurs after 512 bit times, then no retransmission is performed and the end of frame buffer is closed with a Late Collision (LC) error indication.

46.6.16 MII internal and external loopback

Internal and external loopback are supported by the Ethernet controller. In loopback mode, both of the FIFOs are used and the FEC actually operates in a full-duplex fashion. Internal and external loopback are configured using combinations of the RCR[LOOP, DRT] and TCR[FDEN] bits.

Set FDEN for internal and external loopback.

For internal loopback, set RCR[LOOP] and clear RCR[DRT]. FEC_TXEN and FEC_TXER do not assert during internal loopback. During internal loopback, the transmit/receive data rate is higher than in normal operation because the transmit and receive blocks use the internal bus clock instead of the clocks from the external transceiver. This causes an increase in the required system bus bandwidth for transmit

and receive data being DMA'd to/from external memory. It may be necessary to pace the frames on the transmit side and/or limit the size of the frames to prevent transmit FIFO underruns and receive FIFO overflows.

For external loopback, clear RCR[LOOP] and RCR[DRT], and configure the external transceiver for loopback.

46.6.17 RMII loopback

The FEC also supports RMII loopback. In this mode, transmit data is sent to the RMII receive logic and out the FEC_TXD[1:0] pins. To enable RMII loopback mode, set RCR[RMII_MODE, RMII_LOOP] and TCR[FDEN] and clear RCR[RMII_ECHO, LOOP].

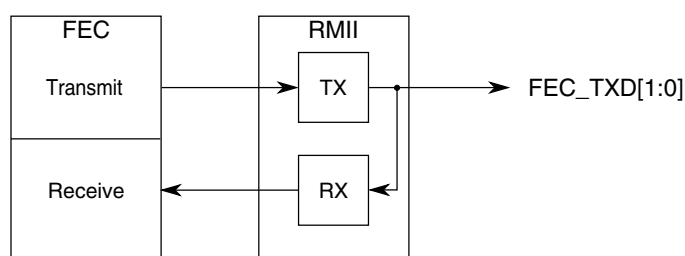


Figure 46-87. RMII loopback mode

46.6.18 RMII echo

The Ethernet controller also supports RMII echo mode, which transmits data on FEC_TXD[1:0] as it is received on FEC_RXD[1:0]. To enable RMII echo mode, set RCR[RMII_MODE, RMII_ECHO] bits and clear RCR[RMII_LOOP, LOOP].

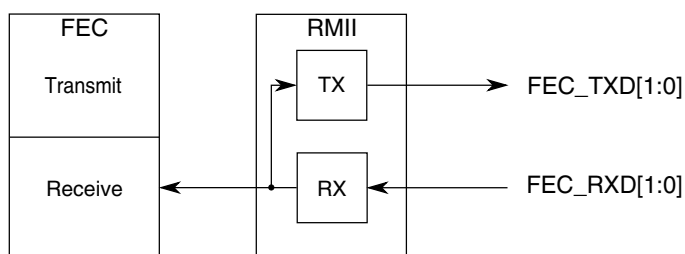


Figure 46-88. RMII echo mode

46.6.19 Ethernet error-managing procedure

The Ethernet controller reports frame reception and transmission error conditions using the MIB block counters, the FEC RxBDs, and the EIR register.

46.6.19.1 Transmission errors

46.6.19.1.1 Transmitter underrun

If this error occurs, the FEC sends 32 bits that ensure a CRC error and stops transmitting. All remaining buffers for that frame are then flushed and closed, and EIR[UN] is set. The FEC then continues to the next transmit buffer descriptor and begin transmitting the next frame. The UN interrupt is asserted if enabled in the EIMR register.

46.6.19.1.2 Retransmission attempts limit expired

When this error occurs, the FEC terminates transmission. All remaining buffers for that frame are flushed and closed, and EIR[RL] is set. The FEC then continues to the next transmit buffer descriptor and begins transmitting the next frame. The RL interrupt is asserted if enabled in the EIMR register.

46.6.19.1.3 Late collision

When a collision occurs after the slot time (512 bits starting at the Preamble), the FEC terminates transmission. All remaining buffers for that frame are flushed and closed, and EIR[LC] is set. The FEC then continues to the next transmit buffer descriptor and begins transmitting the next frame. The LC interrupt is asserted if enabled in the EIMR register.

46.6.19.1.4 Heartbeat

Some transceivers have a self-test feature called heartbeat or signal quality error. To signify a good self-test, the transceiver indicates a collision to the FEC within four microseconds after completion of a frame transmitted by the Ethernet controller. This indication of a collision does not imply a real collision error on the network, but is rather an indication that the transceiver continues to function properly. This is the heartbeat condition.

If TCR[HBC] is set and the heartbeat condition is not detected by the FEC after a frame transmission, a heartbeat error occurs. When this error occurs, the FEC closes the buffer, sets EIR[HB], and generates the HBERR interrupt if it is enabled.

46.6.19.2 Reception errors

46.6.19.2.1 Overrun error

If the receive block has data to put into the receive FIFO and the receive FIFO is full, FEC sets RxBD[OV]. All subsequent data in the frame is discarded and subsequent frames may also be discarded until the receive FIFO is serviced by the DMA and space is made available. At this point the receive frame/status word is written into the FIFO with the OV bit set. The driver must discard this frame.

46.6.19.2.2 Non-octet error (dribbling bits)

The Ethernet controller manages up to seven dribbling bits when the receive frame terminates past a non-octet aligned boundary. Dribbling bits are not used in the CRC calculation. If there is a CRC error, the frame non-octet aligned (NO) error is reported in the RxBD. If there is no CRC error, no error is reported.

46.6.19.2.3 CRC error

When a CRC error occurs with no dribble bits, FEC closes the buffer and sets RxBD[CR]. CRC checking cannot be disabled, but the CRC error can be ignored if checking is not required.

46.6.19.2.4 Frame length violation

When the receive frame length exceeds MAX_FL bytes the BABR interrupt is generated, and RxBD[LG] is set. The frame is not truncated unless the frame length exceeds 2047 bytes.

46.6.19.2.5 Truncation

When the receive frame length exceeds 2047 bytes, frame is truncated and RxBD[TR] is set.

Chapter 47

Peripheral Sensor Interface (PSI5)

47.1 Chip-specific PSI5 information

This chip has two instances of the PSI5 module. The two instances are located on different Peripheral Bridges.

Table 47-1. PSI5 module instances

PSI5 instance	Number of channels	Master or slave operation
PSI5_0	1	Master
PSI5_1	1	Slave

47.1.1 PSI5 GTM interface

The following table identifies the PSI5 connections with eTPU_C, which this chapter refers to as the general timer module (GTM).

Table 47-2. PSI5 GTM connections

Source		Destination		Description
Instance	Pin	Instance	Pin	
eTPU_C	Output channel 14	PSI5_0	GTM reset	eTPU_C channel 14 ¹
eTPU_C	Output channel 13	PSI5_0	GTM event	eTPU_C channel 13
eTPU_C	Output channel 14	PSI5_1	GTM reset	eTPU_C channel 14 ¹
eTPU_C	Output channel 19	PSI5_1	GTM event	eTPU_C channel 19

1. Both PSI5 instances use the same eTPU_C channel.

47.1.2 Detection of PSI5 SMC messages

On this chip, a PSI5 interrupt for received SMC messages is not mapped to an interrupt vector. To detect when each SFR receives and is ready with a new valid SMC message, poll the bits in the GISR[IS_NVSM] field.

47.2 Introduction

The PSI5 interface offers power supply and data transfer between the application and multiple sensors over a two-wire interface. Data transmission from the application to the sensor or sensors is achieved through modulation of the power supply voltage (sync pulse). Data reception from the sensor or sensors to the application is achieved through modulation of the current drawn by a sensor using Manchester encoding.

At the system level, the PSI5 interface consists of mechanisms integrated on the microcontroller and an external physical layer that provides the high-voltage supply and converts the current drawn by the sensor into a digital Manchester-encoded data stream.

47.2.1 Overview

The PSI5 is an interface for automotive sensor applications. PSI5 is an open standard based on existing sensor interfaces for peripheral airbag sensors, already proven in millions of safety airbag systems. The technical characteristics, the low implementation overhead, as well as the attractive cost make the PSI5 also suitable for many other automotive sensor applications. The development goal of the PSI5 is a flexible, reliable communication standard for automotive sensor applications that can be used and implemented free of charge.

The main features of the PSI5 are high-speed and high-reliability data transfer at the lowest possible implementation overhead cost, offering a universal and flexible solution for multiple sensor applications. The main features of PSI5 standard are:

- Two-wire current interface
- Manchester-coded digital data transmission from sensor to ECU
- High data transmission speed of 125 Kbit/s
- High EMC robustness and low emission
- Wide range of sensor supply current
- Configurable data word length (8 to 28 bits)

- Asynchronous or synchronous operation
- Bidirectional communication

In short, PSI5 standardizes the low-level communication between peripheral sensors and electronic control units. Note that the physical layer for the IP (transceiver) is to be implemented separately outside the IP.

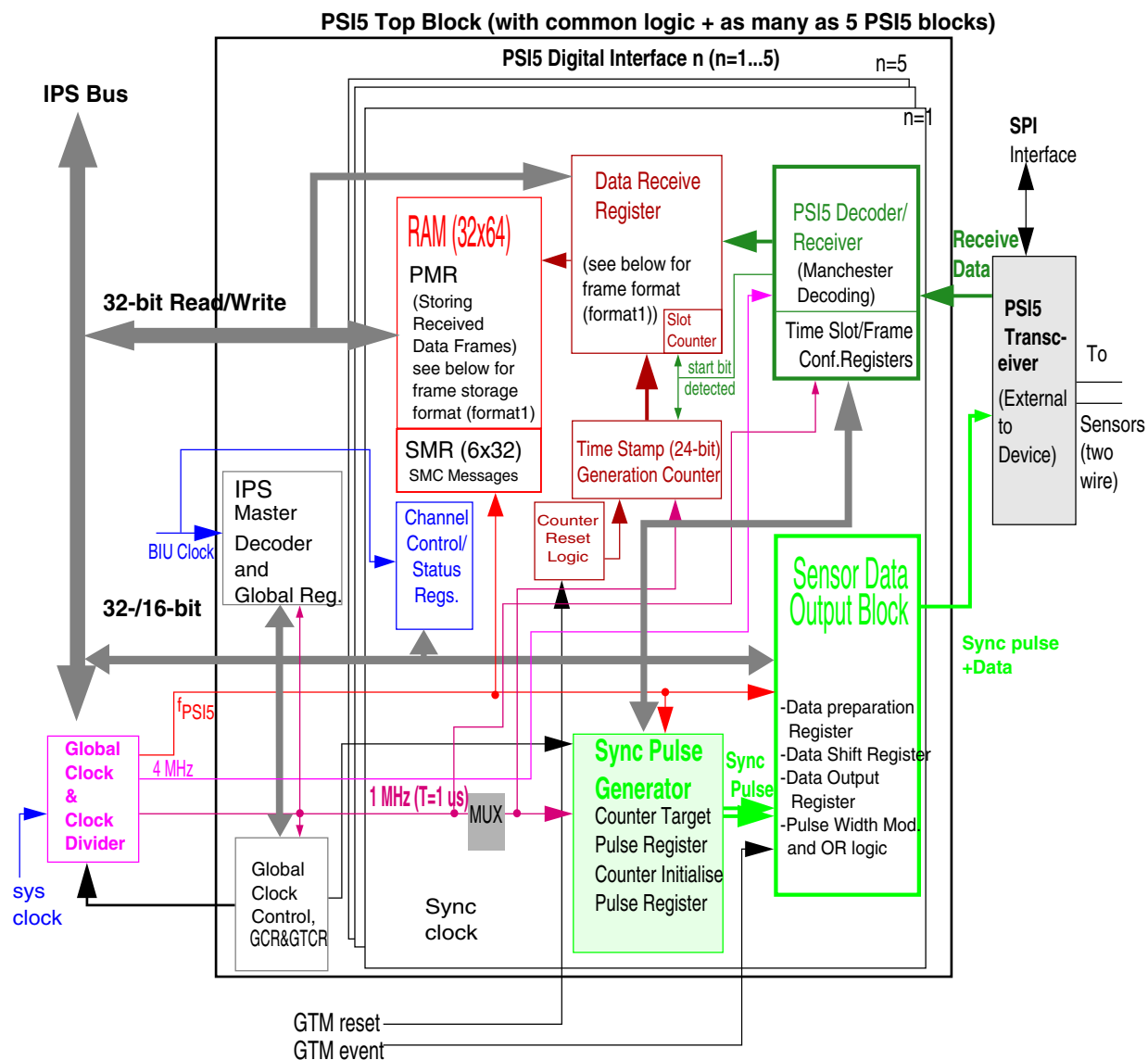


Figure 47-1. PSI5 digital interface block diagram

47.2.2 Features

- IPS interface for register read/write (big endian)

- Supports bidirectional full duplex operations, but with differing baud rate—low, ECU-to-sensor, fast sensor to ECU
- Manchester decoder operating at 125 Kbit/s bit rate with $\times 32$ sample clock rate
- One 64-bit sensor data receive register to which the decoded received sensor data—including header, data, timestamp, frame counter, status bits (E, EM, T, C, F) and CRC, but not including start bits—are written
- A 24-bit time stamp register with a 1 μ s resolution. When first rising edge of start bits (S0) of a message or the rising edge of the SYNC pulse is recognized, the value in the time stamp register will be written to the data receive register
- Provision of resetting the Time Stamp Counter through the reset from the GTM module
- Single 32×64 -bit RAM buffer for storing (moving) content of the data receive register once end of message is recognized
 - Related to each RAM buffer are two diagnostic registers, N and EI. Register N is used to identify the occurrence of new messages in the corresponding RAM buffer, while register EI identifies the error status of these messages. The errors that can set the EI register bits are the F (no frame), E (electrical error), EM (electrical error for SMC bits embedded in PSI5 messages), T (Timing bit error) and C (CRC error). Which of these errors actually set the EI register bits is configurable.
- DMA support for transferring RAM data (including diagnostic data) to system memory
- Support for optional serial messaging channel (SMC) and additional logic to extract SMC frames from PSI5 frames
- Operating modes supported:
 - Asynchronous mode (sensor periodically sends data, no sync pulse, one sensor per interface)
 - Synchronous mode (sync pulse generated, multiple sensor per interface)
- Sync pulse generation for synchronous modes
 - The start of each time slot (1 to 6) (see [Figure 47-128](#)) on the bus (after the end of the sync pulse) can be defined with a 1 μ s resolution

- The end of the last time slot on the bus can be defined with a 1 μ s resolution, up to a maximum sync period time of 16 ms
- Trigger logic to generate sync pulses
 - 16-bit Channel Trigger Counter (CTC) used to generate the sync pulses by comparing the values with CIPR and CTPR
 - Counter Initialize Pulse Register (CIPR) to allow configuration of initialization (offset) period
 - Channel Trigger Period (CTP) register to allow configuration of sync period. Value in CTP is the value to which the CTC is updated after each sync pulse generation
 - Possibility to have a common 1 MHz clock source
- An upstream register bit chain consisting of Preparation Register, Buffer Register, Shift register (output data) and pulse width modulation logic, is implemented. Using this, it is possible to send a data word to the sensors through use of data sync pulses: logic 1 with programmable data sync pulse width W1, logic 0 with programmable data sync pulse width W0. The data sync pulse timing can be configured to be dependent on the periodic or event driven timing SYNC pulse. Further, there is no difference between timing sync pulse and data sync pulse during transmission (ECU-to-sensor communication).

47.2.3 Modes of operation

47.2.3.1 Disable mode

Disable mode in the PSI5 reduces power consumption. This mode is entered by setting GCR[GLOBAL_DISABLE_REQ]. All the channels enter Disable mode at once when this bit is set, irrespective of the setting of PCCR[PSI5_CH_EN] and PCCR[PSI5_CH_CONFIG]. In this mode the RX, TX and common time base counters are disabled for any operation. This is the default state after module reset is released. PSI5 can exit Disable mode by clearing the GLOBAL_DISABLE_REQ bit and enabling individual PSI5 channels (setting PCCR[PSI5_CH_EN]). Note that when entering Disable mode from any other mode, the register contents are retained and are not changed. The time stamp counter AND the internal sync pulse generator are, however, reset when entering Disable mode.

There is no software reset bit in the PSI5 module. The same has to be managed outside the IP at the system level. Channels will enter "Configuration/Normal mode" based on the value of PCCR[PSI5_CH_CONFIG].

47.2.3.2 Initialization/Configuration mode

This is the configuration phase, in which, all PSI5 global, channel specific configuration/control registers can be written. The receiver and transmitter sub-blocks are disabled in this mode. The message receive registers/buffers can be accessed in this mode. This mode can be entered by clearing GCR[GLOBAL_DISABLE_REQ] and setting PCCR[PSI5_CH_CONFIG] and PCCR[PSI5_CH_EN].

47.2.3.3 Normal mode (NORMAL mode)

In Normal mode, the receiver and the transmitter blocks become active, error handling is enabled and all the PSI5 protocol functions are also enabled. In this mode, all the PSI5 global and channel-specific configuration registers are locked for writing, although these registers can still be read. Data Preparation Register, Shift Register and data output registers are writable in this mode. Apart from them, there are some register bits that can be written in this mode also. Please refer to the detailed register description. This mode can be entered clearing GCR[GLOBAL_DISABLE_REQ], clearing PCCR[PSI5_CH_CONFIG], and setting PCCR[PSI5_CH_EN].

47.2.3.3.1 PSI5 topology

The different PSI5 user operation modes define topology and parameters of the communication between ECU and sensors such as communication mode, number of data bits, error detection, cycle time, number of time slots per cycle, and bit rate.

These modes are generally denoted as

PSI5-(A / P / U / D)-dd-(P / CRC)-tttt / n(L / H)

The following figure provides a key for these notations.

Table 47-3. Normal operating states

Communication modes		Description
A	Asynchronous mode	PSI5-A describes a point-to-point connection for unidirectional, asynchronous data transmission. Each sensor is connected to the ECU by two wires. After switching on the power supply, the sensor starts transmitting data to the ECU periodically. Timing and repetition rate of the data transmission are controlled by the sensor. No Sync pulse is send from ECU side.
P	Synchronous Parallel Bus mode	The synchronous operation modes work according to the TDMA method (Time Division Multiple Access). Each sensor is connected to the ECU by two wires. The sensor data transmission is synchronized by the ECU using voltage modulation. Synchronization (sync pulses) can be optionally used for point-to-point configurations; it is mandatory for bus modes. The sensors can be arranged as parallel bus, universal bus and daisy chain topology based on sensor addressing and interchangeability of sensors.
U	Synchronous Universal Bus mode	
D	Synchronous Daisy Chain Bus mode	
Data Bits		
dd	No. of data bits	
Error Detection		
P	1 parity bit	
CRC	3-bit Cyclic Redundancy Check (CRC)	
Cycle Time		
tttt	Cycle Time in μ s	
n	Number of time slots per cycle	
Bit Rate		
L	125 Kbit/s	

47.2.3.4 Debug mode

In order to enter this mode the debugger has to be connected and it has to encounter a breakpoint. In addition, the `DEBUG_EN` bit has to be written to 1. The `DEBUG_EN` bit can be written by either the CPU or by the debugger but only in Configuration mode. Once the above conditions are satisfied the IP enters the Debug Mode. There are two debugging modes available. For simplicity these are being referred to as the “Freeze” and “Open” modes. The IP enters one of the above debugging modes, based on the configuration of a register bit `DEBUG_FREEZE_CTRL` mentioned in the Section 54.3.2.2, PSI5 Channel Control Register (`PSI5_PCCR`).

In Open mode, the core of the IP keeps on working normally and all the registers are updated from the hardware; just like they were being done in the functional mode (when the IP was out of the debug mode). Thus, even a long term entry to the Debug Mode,

won't affect the normal operation of the various registers that are to be updated by the hardware. Down side is that the IP can keep on changing the internal status bits and the registers as the time progresses.

In both the above modes, the various interface registers would be completely visible to the debugger, but debugger action would have no impact on the various status bits and counters that normally get affected by a CPU/DMA read. In this mode, the read pointers of the FIFO do not get incremented, when being read through the Pop-up registers. Thus in Debug mode, reading the DMA PSI5 Message Register (PSI5_DPMR), DMA SMC Frame Register (PSI5_DSFR) or the DMA Diagnostic Status Register (PSI5_DDSR) would keep on returning the same data. Hence the interrupts related to DMA completion operation would not get generated. Further the hardware clearing related to FAST_CLR_SMC and FAST_CLR_PSI5 bits would not have any effect and the hardware clearing would remain disabled. This is also highlighted in the following figure.

47.2.3.5 Operating modes

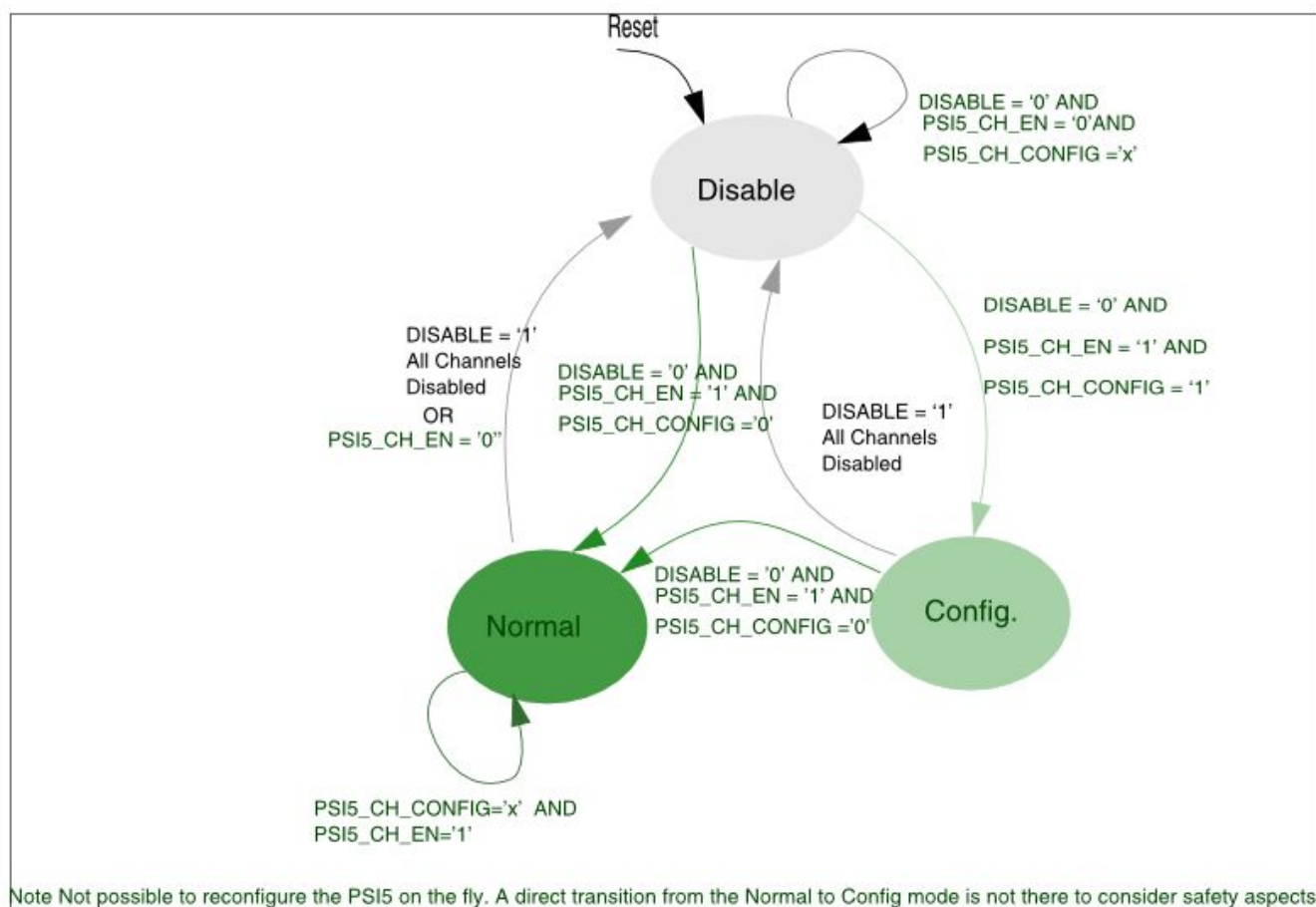


Figure 47-2. Operating modes

Table 47-4. State transition Table

GCR [GLOBAL_DISABLE_REQ]	PCCR		State	
	[PSI5_CH_CONFIG]	[PSI5_CH_EN]	Present	Next
1	x	x	x	All channel Disable
0	x	0	x	Specific channel Disable
0	1	1	Disable	Config
0	0	1	Config	Normal
0	x	1	Normal	Normal
0	0	1	Disable	Normal

47.3 External signal description

External signal description.

Table 47-5. PSI5 port list

Name	Function	I/O	Reset	Pullup
Signals to/from pads				
SDINn_PSI5_m ^{1, 1}	Digital sensor data input pin of PSI5 module	I	0	Active
SDOUTn_PSI5_m ¹	Digital sensor data output pin of PSI5 module (sync pulse + data)	O	0	Passive

1. “n” is the Channel Number and “m” is the Module Instance Number.

47.4 Memory map and register description

The module memory map is shown below.

Unless specifically stated, the register bits are readable in all modes, but writable only in Config mode.

Note that , an access(read/write) to any register in the unimplemented register space will result in the generation of transfer error . All word locations are byte/half word/word accessible for write. Read is always a word access. Trying to do a word operation on 16-bit registers will NOT result in the generation of any transfer error. Trying to write to

"read only" bits of implemented register space, will be ignored without the generation of any error. Trying to write to any register outside its intended mode of write(Normal/config) will be ignored without the generation of any transfer error.

For a summary of the register bit accesses in various device modes, please see [Device modes and Register bit accesses](#)

PSI5 memory map

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
0	Global Control Register (PSI5_GCR)	16	R/W	0000h	47.4.1/2505
8	PSI5 Channel Control Register (PSI5_CH0_PCCR)	32	R/W	001F_0000h	47.4.2/2507
C	DMA Control Register (PSI5_CH0_DCR)	32	R/W	0000_0000h	47.4.3/2513
10	DMA Status Register (PSI5_CH0_DSR)	32	w1c	0000_0000h	47.4.4/2517
14	General Interrupt Control Register (PSI5_CH0_GICR)	32	R/W	0000_0000h	47.4.5/2519
18	New Data Interrupt Control Register (PSI5_CH0_NDICR)	32	R/W	0000_0000h	47.4.6/2521
1C	Overwrite Interrupt Control Register (PSI5_CH0_OWICR)	32	R/W	0000_0000h	47.4.7/2521
20	Error Interrupt Control Register (PSI5_CH0_EICR)	32	R/W	0000_0000h	47.4.8/2522
24	General Interrupt Status Register (PSI5_CH0_GISR)	32	w1c	0007_0000h	47.4.9/2523
28	DMA PSI5 Message Register (PSI5_CH0_DPMR)	32	R	0000_0000h	47.4.10/2525
2C	DMA SMC Frame Register (PSI5_CH0_DSFR)	32	R	0000_0000h	47.4.11/2526
30	DMA Diagnostic Status Register (PSI5_CH0_DDSDR)	32	R	0000_0000h	47.4.12/2526
34	PSI5 Message Receive Register Low (PSI5_CH0_PMRRL)	32	R	0000_0000h	47.4.13/2527
38	PSI5 Message Receive Register High (PSI5_CH0_PMRRH)	32	R	0000_0000h	47.4.14/2527
3C	PSI5 Message Register Low i (PSI5_CH0_PMRL0)	32	R/W	0000_0000h	47.4.15/2529
40	PSI5 Message Register High i (PSI5_CH0_PMRH0)	32	R/W	0000_0000h	47.4.16/2530
44	PSI5 Message Register Low i (PSI5_CH0_PMRL1)	32	R/W	0000_0000h	47.4.15/2529
48	PSI5 Message Register High i (PSI5_CH0_PMRH1)	32	R/W	0000_0000h	47.4.16/2530
4C	PSI5 Message Register Low i (PSI5_CH0_PMRL2)	32	R/W	0000_0000h	47.4.15/2529
50	PSI5 Message Register High i (PSI5_CH0_PMRH2)	32	R/W	0000_0000h	47.4.16/2530
54	PSI5 Message Register Low i (PSI5_CH0_PMRL3)	32	R/W	0000_0000h	47.4.15/2529
58	PSI5 Message Register High i (PSI5_CH0_PMRH3)	32	R/W	0000_0000h	47.4.16/2530

Table continues on the next page...

PSI5 memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
5C	PSI5 Message Register Low i (PSI5_CH0_PMRL4)	32	R/W	0000_0000h	47.4.15/ 2529
60	PSI5 Message Register High i (PSI5_CH0_PMRH4)	32	R/W	0000_0000h	47.4.16/ 2530
64	PSI5 Message Register Low i (PSI5_CH0_PMRL5)	32	R/W	0000_0000h	47.4.15/ 2529
68	PSI5 Message Register High i (PSI5_CH0_PMRH5)	32	R/W	0000_0000h	47.4.16/ 2530
6C	PSI5 Message Register Low i (PSI5_CH0_PMRL6)	32	R/W	0000_0000h	47.4.15/ 2529
70	PSI5 Message Register High i (PSI5_CH0_PMRH6)	32	R/W	0000_0000h	47.4.16/ 2530
74	PSI5 Message Register Low i (PSI5_CH0_PMRL7)	32	R/W	0000_0000h	47.4.15/ 2529
78	PSI5 Message Register High i (PSI5_CH0_PMRH7)	32	R/W	0000_0000h	47.4.16/ 2530
7C	PSI5 Message Register Low i (PSI5_CH0_PMRL8)	32	R/W	0000_0000h	47.4.15/ 2529
80	PSI5 Message Register High i (PSI5_CH0_PMRH8)	32	R/W	0000_0000h	47.4.16/ 2530
84	PSI5 Message Register Low i (PSI5_CH0_PMRL9)	32	R/W	0000_0000h	47.4.15/ 2529
88	PSI5 Message Register High i (PSI5_CH0_PMRH9)	32	R/W	0000_0000h	47.4.16/ 2530
8C	PSI5 Message Register Low i (PSI5_CH0_PMRL10)	32	R/W	0000_0000h	47.4.15/ 2529
90	PSI5 Message Register High i (PSI5_CH0_PMRH10)	32	R/W	0000_0000h	47.4.16/ 2530
94	PSI5 Message Register Low i (PSI5_CH0_PMRL11)	32	R/W	0000_0000h	47.4.15/ 2529
98	PSI5 Message Register High i (PSI5_CH0_PMRH11)	32	R/W	0000_0000h	47.4.16/ 2530
9C	PSI5 Message Register Low i (PSI5_CH0_PMRL12)	32	R/W	0000_0000h	47.4.15/ 2529
A0	PSI5 Message Register High i (PSI5_CH0_PMRH12)	32	R/W	0000_0000h	47.4.16/ 2530
A4	PSI5 Message Register Low i (PSI5_CH0_PMRL13)	32	R/W	0000_0000h	47.4.15/ 2529
A8	PSI5 Message Register High i (PSI5_CH0_PMRH13)	32	R/W	0000_0000h	47.4.16/ 2530
AC	PSI5 Message Register Low i (PSI5_CH0_PMRL14)	32	R/W	0000_0000h	47.4.15/ 2529
B0	PSI5 Message Register High i (PSI5_CH0_PMRH14)	32	R/W	0000_0000h	47.4.16/ 2530

Table continues on the next page...

PSI5 memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
B4	PSI5 Message Register Low i (PSI5_CH0_PMRL15)	32	R/W	0000_0000h	47.4.15/2529
B8	PSI5 Message Register High i (PSI5_CH0_PMRH15)	32	R/W	0000_0000h	47.4.16/2530
BC	PSI5 Message Register Low i (PSI5_CH0_PMRL16)	32	R/W	0000_0000h	47.4.15/2529
C0	PSI5 Message Register High i (PSI5_CH0_PMRH16)	32	R/W	0000_0000h	47.4.16/2530
C4	PSI5 Message Register Low i (PSI5_CH0_PMRL17)	32	R/W	0000_0000h	47.4.15/2529
C8	PSI5 Message Register High i (PSI5_CH0_PMRH17)	32	R/W	0000_0000h	47.4.16/2530
CC	PSI5 Message Register Low i (PSI5_CH0_PMRL18)	32	R/W	0000_0000h	47.4.15/2529
D0	PSI5 Message Register High i (PSI5_CH0_PMRH18)	32	R/W	0000_0000h	47.4.16/2530
D4	PSI5 Message Register Low i (PSI5_CH0_PMRL19)	32	R/W	0000_0000h	47.4.15/2529
D8	PSI5 Message Register High i (PSI5_CH0_PMRH19)	32	R/W	0000_0000h	47.4.16/2530
DC	PSI5 Message Register Low i (PSI5_CH0_PMRL20)	32	R/W	0000_0000h	47.4.15/2529
E0	PSI5 Message Register High i (PSI5_CH0_PMRH20)	32	R/W	0000_0000h	47.4.16/2530
E4	PSI5 Message Register Low i (PSI5_CH0_PMRL21)	32	R/W	0000_0000h	47.4.15/2529
E8	PSI5 Message Register High i (PSI5_CH0_PMRH21)	32	R/W	0000_0000h	47.4.16/2530
EC	PSI5 Message Register Low i (PSI5_CH0_PMRL22)	32	R/W	0000_0000h	47.4.15/2529
F0	PSI5 Message Register High i (PSI5_CH0_PMRH22)	32	R/W	0000_0000h	47.4.16/2530
F4	PSI5 Message Register Low i (PSI5_CH0_PMRL23)	32	R/W	0000_0000h	47.4.15/2529
F8	PSI5 Message Register High i (PSI5_CH0_PMRH23)	32	R/W	0000_0000h	47.4.16/2530
FC	PSI5 Message Register Low i (PSI5_CH0_PMRL24)	32	R/W	0000_0000h	47.4.15/2529
100	PSI5 Message Register High i (PSI5_CH0_PMRH24)	32	R/W	0000_0000h	47.4.16/2530
104	PSI5 Message Register Low i (PSI5_CH0_PMRL25)	32	R/W	0000_0000h	47.4.15/2529
108	PSI5 Message Register High i (PSI5_CH0_PMRH25)	32	R/W	0000_0000h	47.4.16/2530

Table continues on the next page...

PSI5 memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
10C	PSI5 Message Register Low i (PSI5_CH0_PMRL26)	32	R/W	0000_0000h	47.4.15/2529
110	PSI5 Message Register High i (PSI5_CH0_PMRH26)	32	R/W	0000_0000h	47.4.16/2530
114	PSI5 Message Register Low i (PSI5_CH0_PMRL27)	32	R/W	0000_0000h	47.4.15/2529
118	PSI5 Message Register High i (PSI5_CH0_PMRH27)	32	R/W	0000_0000h	47.4.16/2530
11C	PSI5 Message Register Low i (PSI5_CH0_PMRL28)	32	R/W	0000_0000h	47.4.15/2529
120	PSI5 Message Register High i (PSI5_CH0_PMRH28)	32	R/W	0000_0000h	47.4.16/2530
124	PSI5 Message Register Low i (PSI5_CH0_PMRL29)	32	R/W	0000_0000h	47.4.15/2529
128	PSI5 Message Register High i (PSI5_CH0_PMRH29)	32	R/W	0000_0000h	47.4.16/2530
12C	PSI5 Message Register Low i (PSI5_CH0_PMRL30)	32	R/W	0000_0000h	47.4.15/2529
130	PSI5 Message Register High i (PSI5_CH0_PMRH30)	32	R/W	0000_0000h	47.4.16/2530
134	PSI5 Message Register Low i (PSI5_CH0_PMRL31)	32	R/W	0000_0000h	47.4.15/2529
138	PSI5 Message Register High i (PSI5_CH0_PMRH31)	32	R/W	0000_0000h	47.4.16/2530
13C	SMC Frame Register n (PSI5_CH0_SFR1)	32	R/W	0000_0000h	47.4.17/2531
140	SMC Frame Register n (PSI5_CH0_SFR2)	32	R/W	0000_0000h	47.4.17/2531
144	SMC Frame Register n (PSI5_CH0_SFR3)	32	R/W	0000_0000h	47.4.17/2531
148	SMC Frame Register n (PSI5_CH0_SFR4)	32	R/W	0000_0000h	47.4.17/2531
14C	SMC Frame Register n (PSI5_CH0_SFR5)	32	R/W	0000_0000h	47.4.17/2531
150	SMC Frame Register n (PSI5_CH0_SFR6)	32	R/W	0000_0000h	47.4.17/2531
154	New Data Status Register (PSI5_CH0_NDSR)	32	w1c	0000_0000h	47.4.18/2533
158	Overwrite Status Register (PSI5_CH0_OWSR)	32	w1c	0000_0000h	47.4.19/2533
15C	Error Indication Status Register (PSI5_CH0_EISR)	32	w1c	0000_0000h	47.4.20/2534
160	Set New Data Status Register (PSI5_CH0_SNDSR)	32	W	0000_0000h	47.4.21/2534

Table continues on the next page...

PSI5 memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
164	Set Overwrite Status Register (PSI5_CH0_SOWSR)	32	W	0000_0000h	47.4.22/ 2535
168	Set Error Status Register (PSI5_CH0_SEISR)	32	W	0000_0000h	47.4.23/ 2535
16C	Set SMC Error Status Register (PSI5_CH0_SSESR)	32	W	0000_0000h	47.4.24/ 2536
170	Sync Time Stamp Read Register (PSI5_CH0_STSR)	32	R	0000_0000h	47.4.25/ 2537
174	Data Time Stamp Read Register (PSI5_CH0_DTSRR)	32	R	0000_0000h	47.4.26/ 2537
178	Slot n Frame Configuration Register (PSI5_CH0_S1FCR)	32	R/W	0000_0010h	47.4.27/ 2538
17C	Slot n Frame Configuration Register (PSI5_CH0_S2FCR)	32	R/W	0000_0010h	47.4.27/ 2538
180	Slot n Frame Configuration Register (PSI5_CH0_S3FCR)	32	R/W	0000_0010h	47.4.27/ 2538
184	Slot n Frame Configuration Register (PSI5_CH0_S4FCR)	32	R/W	0000_0010h	47.4.27/ 2538
188	Slot n Frame Configuration Register (PSI5_CH0_S5FCR)	32	R/W	0000_0010h	47.4.27/ 2538
18C	Slot n Frame Configuration Register (PSI5_CH0_S6FCR)	32	R/W	0000_0010h	47.4.27/ 2538
190	Slot 1 Start Boundary Register (PSI5_CH0_S1SBR)	16	R	0000h	47.4.28/ 2540
192	Slot 2 Start Boundary Register (PSI5_CH0_S2SBR)	16	R	0000h	47.4.29/ 2540
194	Slot 3 Start Boundary Register (PSI5_CH0_S3SBR)	16	R	0000h	47.4.30/ 2541
196	Slot 4 Start Boundary Register (PSI5_CH0_S4SBR)	16	R	0000h	47.4.31/ 2541
198	Slot 5 Start Boundary Register (PSI5_CH0_S5SBR)	16	R	0000h	47.4.32/ 2542
19A	Slot 6 Start Boundary Register (PSI5_CH0_S6SBR)	16	R	0000h	47.4.33/ 2542
19C	Slot n End Boundary Register (PSI5_CH0_SnEBR)	32	R/W	0000_0000h	47.4.34/ 2543
1A0	Data Output Block Configuration Register (PSI5_CH0_DOBCR)	16	R/W	0000h	47.4.35/ 2544
1A2	Manchester Decoder Disable Offset (PSI5_CH0_MDDIS_OFF)	16	R/W	0000h	47.4.36/ 2547
1A4	Pulse Width for Data Bit Value 0 (PSI5_CH0_PW0D)	16	R/W	0000h	47.4.37/ 2548
1A6	Pulse Width for Data Bit Value 1 (PSI5_CH0_PW1D)	16	R/W	0000h	47.4.38/ 2548

Table continues on the next page...

PSI5 memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
1A8	Counter Target Pulse Register (PSI5_CH0_CTPR)	16	R/W	0000h	47.4.39/2549
1AA	Counter Initialize Pulse Register (PSI5_CH0_CIPR)	16	R/W	0000h	47.4.40/2550
1AC	Data Preparation Register Low (PSI5_CH0_DPRL)	32	R/W	0000_0000h	47.4.41/2550
1B0	Data Preparation Register High (PSI5_CH0_DPRH)	32	R	0000_0000h	47.4.42/2551
1B4	Data Buffer Register Low (PSI5_CH0_DBRL)	32	R/W	0000_0000h	47.4.43/2552
1B8	Data Buffer Register High (PSI5_CH0_DBRH)	32	R/W	0000_0000h	47.4.44/2554
1BC	Data Shift Register Low (PSI5_CH0_DSRL)	32	R/W	0000_0000h	47.4.45/2556
1C0	Data Shift Register High (PSI5_CH0_DSRH)	32	R/W	0000_0000h	47.4.46/2557

47.4.1 Global Control Register (PSI5_GCR)

The Global Control Register (PSI5_GCR)¹ provides control functions for all the PSI5 channels simultaneously.

Address: 0h base + 0h offset = 0h

Bit	0	1	2	3	4	5	6	7
Read	0							
Write								
Reset	0	0	0	0	0	0	0	0
Bit	8	9	10	11	12	13	14	15
Read	0						CTC_GED	GLOBAL_DISABLE_REQ
Write								
Reset	0	0	0	0	0	0	0	0

PSI5_GCR field descriptions

Field	Description
0–13 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
14 CTC_GED	Channel Target Counter (CTC) Global Enable/Disable. NOTE: This bit is writable in all modes.

Table continues on the next page...

PSI5_GCR field descriptions (continued)

Field	Description
	0 All the CTC counters having PCCR[CTC_GED_SEL] = 1 are reset. 1 Simultaneously starts the CTC of all the channels that have PCCR[CTC_GED_SEL] = 1.
15 GLOBAL_DISABLE_REQ	PSI5 Receive Global disable request. When set, all PSI5 channels enter Disable mode (from individual Normal or Config modes) regardless of the settings of the PCCR[PSI5_CH_CONFIG] and PCCR[PSI5_CH_EN] bits. When cleared, each PSI5 channel enters Config mode or Normal mode based on the setting of the PCCR[PSI5_CH_CONFIG] bit if PCCR[PSI5_CH_CONFIG] is set for that channel; otherwise, that particular channel continues to stay in Disable mode. The reset value is dependent on the parameter PSI5_PDIS_RST. When PSI5_PDIS_RST = 0, then the reset value is "0" (PSI5 module is enabled after reset). Note that when GLOBAL_DISABLE_REQ == 1, then all the clock enables (shown in Figure 1) from the IP go to a 0 (inactive) state. This bit is writable in all the modes. 0 PSI5 channel(s) enters Normal or Config mode, depending on the value of the PCCR[PSI5_CH_CONFIG] bit. 1 All PSI5 Channels are in Disable mode.

1. This register is not present if the specific instance is configured as a slave. Master or slave configuration of the specific PSI5 instance decides if the operating mode (DISABLE/CONFIG/NORMAL) as well as the staggering of the channels can be controlled independently. If the instance is configured as a master, then it controls its own operating mode, staggering of all the channels in that instance and the operating mode and the staggering of any associated slave instance. If the instance is configured as a slave, then its operating mode and staggering can only be controlled by another master instance. Please refer to the Device Configuration Chapter for instance specific details.

47.4.2 PSI5 Channel Control Register (PSI5_CH0_PCCR)

This section shows the PSI5 Channel Control Register.

Address: 0h base + 8h offset = 8h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	CTC_GED_SEL	CTC_ED	0	MEM_DEPTH					0			ERROR_SELECT4	ERROR_SELECT3	ERROR_SELECT2	ERROR_SELECT1	ERROR_SELECT0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	
R	Reserved	GTM_RESET_ASYNC_EN	0				DEBUG_EN	DEBUG_FREEZE_CTRL	SP_TS_CLK_SEL	0		FAST_CLR_SMC	FAST_CLR_PSI5	BIT_RATE	MODE	PSI5_CH_CONFIG	PSI5_CH_EN
W																	
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	

PSI5_CH0_PCCR field descriptions

Field	Description
0 CTC_GED_SEL	Channel Target Counter, Global Enable/Disable Select. This bit selects whether the CTC should be enabled/disabled by CTC_GED bit (in GCR) or the CTC_ED bit. Enabling by the CTC_GED bit is used when staggering of the channels is required. NOTE: This bit is writable only in Config mode. 0 CTC enabled disabled by CTC_ED. 1 CTC enabled/disabled by CTC_GED.
1 CTC_ED	Channel Target Counter Enable/Disable. This bit is used to control the CTC locally from the channel in case the CTC_GED_SEL = 0. NOTE: This bit is writable only in Normal mode and the Config mode. 0 The CTC counters is disabled and reset. 1 The CTC counter is enabled and start counting.
2 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
3–7 MEM_DEPTH	Can be programmed from 0 to 31 and denotes the size of the memory that should be used for storing the PSI5 messages. Area above the MEM_DEPTH is treated as being unavailable for message storage.

Table continues on the next page...

PSI5_CH0_PCCR field descriptions (continued)

Field	Description
	MEM_DEPTH also denotes the number of RAM registers available for the CPU access. Note that the same memory can be accessed sequentially (though the DMA) or randomly anytime by the CPU. 0: it indicates that only 1 location of the memory is available for storage 31: it indicates that all the 32 locations of the memory are available for storage. Notes: - Depending on the configuration of the DMA_PM_DS_CONFIG bits in the DCR, the MEM_DEPTH memory size may be accessed by CPU only (interrupts are generated) or CPU and DMA (through the dedicated DMA popup registers). - DMA_PM_DS_CONFIG = conf1 then DMA requests are disabled and only the interrupts can be used for data transfer. - DMA_PM_DS_CONFIG = conf2,conf3,conf4 then DMA requests are enabled. The interrupts should be disabled by writing a 0 in the corresponding IE bits. - For more details please refer to description of DMA_PM_DS_CONFIG in the DCRs. - The data in the memory area defined by the MEM_DEPTH parameter is always filled in a sequential fashion as in a FIFO, but can be read sequentially or randomly. There can be no slot-memory location correspondence due to the T bit errors in the manchester decoder. - The memory area above the configured MEM_DEPTH size will always be read as 0. - Irrespective of the configuration, the data can always be randomly read by the CPU from the PMRL/PMRH register. NOTE: This bit is writable only in Config mode.
8–10 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
11 ERROR_SELECT4	The ERROR_SELECT bitfield indicates which of the C, E, EM, T, or F error conditions generate an interrupt when the EICR Interrupt Enable bit is set. The individual bits are mapped as {C,E,EM,T,F}. The corresponding error is also latched in the PSI5_EISR. Default: ERROR_SELECT = 11111: Any of C,E,EM,T,or F generate an interrupt. C error CRC Error (C) Bit 20: ERROR_SELECT[4] or C_INT_SEL NOTE: This bit is writable only in Config mode. 0 The C bit does not generate an interrupt. 1 The C bit generates an interrupt.
12 ERROR_SELECT3	Error_Select[3] or E_INT_SEL E error Electrical Error (E) NOTE: This bit is writable only in Config mode. 0 The E bit does not generate an interrupt. 1 The E bit generates an interrupt.
13 ERROR_SELECT2	Error_Select[2] or EM_INT_SEL NOTE: This bit is writable only in Config mode.

Table continues on the next page...

PSI5_CH0_PCCR field descriptions (continued)

Field	Description
	0 The EM bit does not generate an interrupt. 1 The EM bit generates an interrupt.
14 ERROR_SELECT1	Error_Select[1] or T_INT_SEL NOTE: This bit is writable only in Config mode. 0 The T bit does not generate an interrupt. 1 The T bit generates an interrupt.
15 ERROR_SELECT0	Error_Select[0] or F_INT_SEL NOTE: This bit is writable only in Config mode. 0 The F bit does not generate an interrupt. 1 The F bit generates an interrupt.
16 Reserved	This field is reserved.
17 GTM_RESET_ASYNC_EN	GTM_RESET_ASYNC_EN NOTE: This bit is writable only in Config mode. 0 Both the assertion and the deassertion of the GTM reset are treated synchronously. Inside the IP the gtm reset is synchronized on the SP_TS_CLK. 1 Assertion of the GTM reset is treated asynchronously but deassertion is synchronized on the SP_TS_CLK.(For details of SP_TS_CLK, refer to bit 23 of) PSI5 Channel Control Register (PSI5_CH0_PCCR)
18–20 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
21 DEBUG_EN	This bit allows/prevents the IP from entering the debug mode, whenever the debugger is connected and a breakpoint is encountered by the debugger: NOTE: This bit is writable only in Config mode. 0 The IP never enters the debug mode, even if the debugger is connected and a breakpoint is encountered. It keeps on working normally, as in the functional mode. 1 Whenever the debugger is connected and a breakpoint is encountered, the IP enters the debug mode. In the Debug mode, the DMA read pointers and other status registers that normally get affected by the READ operation remain unaffected.
22 DEBUG_FREEZE_CTRL	DEBUG_FREEZE_CTRL NOTE: This bit is writable only in Config mode.

Table continues on the next page...

PSI5_CH0_PCCR field descriptions (continued)

Field	Description
	0 When the IP enters the Debug mode then it goes into the “Open” mode, i.e., the core of the IP keeps on working normally. In this case, there can be a difference between the value of a particular status bit, which is there at the time Debug mode is entered and the time at which the debugger reads it, since some upcoming events might change the status. The hardware clearing, of course, remains disabled. NOTE: In the above configuration, the status of registers read through the debugger may be different when entering debug mode and when actually accessing registers at some later point of time. 1 As soon as debug mode is asserted, the state of the system registers that are affected by the hardware gets frozen to what ever it was existing at the point at which the Debug mode goes active. This can be useful for reading the state of the system at a particular point of time such that it is NOT affected by the next upcoming events. Note that Manchester decoder/ Sync Pulse Generator/ Time Stamp counters keep on working normally but the registers (data/status/timestamp) affected by these are not updated. When the IP enters the “Debug mode” then it goes into the “Freeze Mode”. Note that whenever there is a request to enter the freeze mode then the IP enters this mode, ONLY when the Manchester decoder and the Sync Pulse Generator are stopped coherently. This means that before entering the “freeze mode”, if there is any pending reception that is ongoing, then the same is completed first. Simultaneously the “sdout” line of the IP should be at a low level. Once these two conditions are satisfied, the IP enters the Debug Freeze mode. The Time Stamp counters are also halted. The fact that the freeze mode has been entered, can be checked by reading the GISR[IS_DB_FR] bit. Once the IP comes out of the freeze mode then this bit goes to 0. NOTE: In the above configuration, incoming packets could get lost while the Debug mode is active.
23 SP_TS_CLK_SEL	This bit controls which clock goes to the Sync Pulse and Time Stamp Generator Unit. NOTE: The above selection affects ONLY the clock for the Time Stamp Unit and Sync Pulse Generator. The 1 MHz clock to the rest of the logic like the Manchester decoder or the Slot Boundary Counters is still the 1 MHz from the common clock generator module. NOTE: The clock selected using this bit is called SP_TS_CLK. 0 The 1 MHz clock generated from the common clock generator module goes for clocking the Time Stamp Unit and sync pulse generator. 1 Reserved.
24–25 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
26 FAST_CLR_SMC	This bit controls the clearing mechanism of the IS_NVSM[x], IS_CESM[x], and IS_OWSM[x] bits in the GISR. NOTE: This bit is effective only when DCR[DMA_EN_SF] = 1. When FAST_CLR_SMC = 1 then the IS_NVSM[x], IS_CESM[x], and IS_OWSM[x] bits are automatically cleared upon reading the DSFR message register, as per the slot[x] message read in from the DSFR. When reading directly from the SFR registers, this bit does not affect auto clearing. If FAST_CLR_SMC = 0 then the above bits have to be cleared by a w1c. NOTE: This bit is writable only in Config mode. 0 Fast clearing is disabled (clear when written to 1). 1 Fast clearing is enabled.
27 FAST_CLR_PSI5	This bit control the clearing mechanism of bits in NDSR(in conf2, conf3 and conf4 modes), EISR (in conf2, conf3 and conf4 modes), and OWSR(in conf2 and conf3 modes). NOTE: This bit is effective only when DMA_PM_DS_CONFIG = conf2, conf3, or conf4 else this bit has no effect and only software clearing is available. Further, though the OWSR is not directly read by

Table continues on the next page...

PSI5_CH0_PCCR field descriptions (continued)

Field	Description
	the DMA as a single register, it is cleared in specific modes (when FAST_CLR_PSI5 == 1) as can be seen below. Following are the details in the various modes of the DCR[DMA_PM_DS_CONFIG] bits: • In conf1 mode: This bit has no effect. Reading the DDSR and the DPMR returns a 0. • In conf2 mode: In this case the diagnostic bits (EISR, NDSR) are cleared ONLY when the diagnostic registers appended with the corresponding PSI5 messages are completely read using the DPMR. During the intermediate operation (when the PSI5 message has been read but the diagnostic bits are still pending), the PSI5 message read from the DPMR will not clear these bits. Both of these registers are simultaneously cleared when the corresponding DMA operation is over. However, only those bits are cleared, the corresponding messages of which form the part of the DMA request being serviced. Note that the OWSR bits are cleared upon the corresponding PSI5 message read from the DPMR. Reading the DDSR returns a 0 in this mode. • In conf3 mode: The PSI5 diagnostic register bits (EISR,NDSR) and OWSR are automatically cleared when the corresponding PSI5 message is read from the DPMR. The specific register bits of EISR,NDSR and OWSR, corresponding to each message, are cleared sequentially one by one as the corresponding message is being read by the DMA. Note that the three bits (each of EISR, NDSR, and OWSR) corresponding to a specific location are always cleared simultaneously. Reading the DDSR returns a 0. When reading from the PMRH/PMRL registers, this bit does not have any effect on fast clearing. • In conf4 mode: In this case the diagnostic bits (EISR, NDSR) are cleared as soon as these registers are read through the DDSR. These registers are cleared simultaneously in one go, as soon as the corresponding DMA request has been serviced. Only those bits of these registers are cleared, the messages of which form a part of the DMA request. Note that in this mode the reading of the PSI5 messages from the DMA is not available as the request is being used for reading the diagnostic registers. Reading the DPMR returns a 0. The PSI5 messages can be read only from the PMRH/PMRL register. Hence the OWSR has to be cleared only by w1c. • In all the above modes, when reading from the PMRH/PMRL registers, the FAST_CLR_PSI5 does not have any effect on fast clearing. • Further, In all the above cases if FAST_CLR_PSI5 = 0 then the specific diagnostic bits and the OWSR have to be cleared by w1c. NOTE: This bit is writable only in Config mode. 1 Fast Clearing Enabled 0 Fast Clearing Disables (clear when written to 1)
28 BIT_RATE	Bit Rate This bit selects the receive message bit rate (T bit) for this particular PSI5 Channel, that is, it selects the 4 MHz driven clock in the common clock generator module. NOTE: This bit is writable only in Config mode. 0 125 Kbit/s bit rate selected (4 MHz clock). 1 Reserved
29 MODE	This bit selects the operating modes. NOTE: This bit is writable only in Config mode.

Table continues on the next page...

PSI5_CH0_PCCR field descriptions (continued)

Field	Description
	0 Asynchronous operating mode (Integrated sync pulse generator module is switched off, only RX active). Note that in this mode the sync pulses from the internal sync pulse generator or the GTM are not allowed to be propagated to the output path, which always remains at logic 0. There is no T-bit error in the Asynchronous mode. 1 Synchronous operating modes (Integrated sync pulse generator module is switched on, both TX and RX subblocks active).
30 PSI5_CH_CONFIG	PSI5 Channel Config mode request. If this bit is set (and PCCR[PSI5_CH_EN] = '1') this particular psi5 channel enters Configuration mode from Disable mode. Default value is 0. User software must program all PSI5 channel parameters before clearing this bit. Once cleared, channel parameters in registers are locked for writing. The values set can not be changed until this bit is set again after entering the Disable mode NOTE: This bit can be written in all modes, but writing a 0 in the Normal mode does not change the operation mode of the device. For more details please see Figure 47-2 and Table 47-4. Do note that when going from Disable to Config mode, the programming of other bits of this register CANNOT be done in the same cycle, as the one in which this bit is being written. 1 (If PCCR[PSI5_CH_EN] = '1' and GCR[GLOBAL_DISABLE_REQ] = '0') - PSI5 Channel enters Config mode. Various configuration registers can be programmed in this mode. The only registers NOT writable in the config mode but writable in the Normal mode are the output registers PSI5_DBR and the PSI5_DSR. 0 (If PCCR[PSI5_CH_EN] = '1' and GCR[GLOBAL_DISABLE_REQ] = '0') PSI5 Channel enters Normal mode. Channel parameter registers are locked for writing. This bit can be written in all the modes but writing a "0" in the Normal mode will not change the operation mode of the device.
31 PSI5_CH_EN	PSI5 Channel Enable. If this bit is cleared this particular psi5 channel continues to stay in Disable mode even if GCR[GLOBAL_DISABLE_REQ] = 0. If set, it enters Configuration mode from Disable mode (based on PCCR[PSI5_CH_CONFIG]). Default value is 0. All state machines are disabled. This bit can be written in all modes. NOTE: When PSI5_CH_EN == 0, then the clock enable for that particular channel that goes from the IP, goes to "0" (inactive state). This also means that the only clock available to the IP in the config mode is the "ipg_clk_s". Do note that when going from Disable to Config mode, the programming of other bits of this register CANNOT be done in the same cycle, as the one in which this bit is being written. 0 PSI5 channel continues in Disable mode. 1 PSI5 channel enabled to enter Config/Normal mode.

47.4.3 DMA Control Register (PSI5_CH0_DCR)

This section defines the DMA Control Register.

Address: 0h base + Ch offset = Ch

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	Reserved	0									0			IE_DMA_TF_SF	IE_DMA_TF_PM_DS	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R		0									0					
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

PSI5_CH0_DCR field descriptions

Field	Description
0 Reserved	This field is reserved. NOTE: Only write this field with the reset value.
1–2 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
3–7 DMA_PM_DS_WM	0 to 31: Valid water mark levels. Value fixed by user software. Since each location has two 32-bit registers hence a total of sixty-four 32-bit words are present. The default and minimum value of dma_pm_ds_wm is '0', i.e. 1 message/Diagnostic bit of 1 location. The maximum value of dma_pm_ds_wm is 31, i.e. the actual watermark = 32 RAM locations. NOTE: - These bits are writable only in config mode. - These bits are ineffective if DMA_PM_DS_CONFIG = conf1. - These bits indicates the number of PSI5 message to be stored in FIFO / number of unread diagnostic bits in the diagnostic registers with a correspondence to the messages in the FIFO, before the corresponding DMA request is asserted. The watermark definition changes depending on the configuration of the DMA_PM_DS_CONFIG bits in the DCR as mentioned below.

Table continues on the next page...

PSI5_CH0_DCR field descriptions (continued)

Field	Description
	- When DMA_PM_DS_CONFIG = conf2 or conf3 then the watermark refers to the number of new unread PSI5 messages that are stored in the FIFO before a DMA request is asserted. - In conf2 mode of DMA_PM_DS_CONFIG, the number of words to be transferred, set in the DMA controllers TCD must be set equal to the number of words corresponding to the set water mark value + 2 words (for the two 32-bit diagnostic registers). - In conf3, the number of words to be transferred set in the DMA controllers TCD must be set equal to the number of words corresponding to the set water mark value. - When DMA_PM_DS_CONFIG = conf4 then the watermark refers to the number of new unread diagnostic bits in the diagnostic register. In this configuration the PSI5 messages can only be read by the CPU, since the DMA request would cater to reading the diagnostic bits only. - In conf4, the number of words transferred in each DMA request = 2 x 32-bit words (for the two 32-bit diagnostic registers). The water mark however refers to the number of new unread diagnostic bits pertaining to each PSI5 message in the FIFO.
8–12 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
13 IE_DMA_TF_SF	NOTE: This bit is writeable in Config and Normal modes. Enable for interrupt, which is generated when DMA transfer finishes for DMA SMC Frame register. 0 Interrupt is disabled. 1 Interrupt is enabled.
14 IE_DMA_TF_PM_DS	Enable for Interrupt, which is generated when DMA transfer finishes for PSI5 messages/DMA Diagnostic Status register depending on the DMA_PM_DS_CONFIG bits of the DCR. NOTE: This bit is writable only in Config and Normal modes. 0 Interrupt is disabled. 1 Interrupt is enabled.
15–19 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
20 IE_DMA_PM_DS_FIFO_FULL	This bit is effective only when the DMA_PM_DS_CONFIG = conf2, conf3 or conf4. It enables the FIFO FULL condition generation interrupt. For the cases which generate these FIFO FULL conditions please refer to the description of IS_DMA_PM_DS_FIFO_FULL register bit of the DSR. NOTE: This bit is writable only in Config and Normal modes.
21 IE_DMA_SFUF	Enables interrupt when there is underflow in DMA SMC Frame register when DMA is enabled. It is set when the DSFR is read without a valid DMA request, i.e. it is empty. Default value is 0. NOTE: This bit is writable only in Config and Normal modes. 0 Interrupt is disabled. 1 Interrupt is enabled.
22 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

Table continues on the next page...

PSI5_CH0_DCR field descriptions (continued)

Field	Description
23 IE_DMA_PM_DS_UF	Enables interrupt when there is underflow in DMA PSI5 message register when DMA is enabled. Default value is 0. For the details as to when this underflow occurs, please refer to the details of IS_DMA_PM_DS_UF bit details of the DSR. NOTE: This bit is writable in Config and Normal modes. 0 Interrupt is disabled. 1 Interrupt is enabled.
24–28 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
29 DMA_EN_SF	Enable DMA request for SMC Frame Data NOTE: This bit is writable only in config mode. 0 DMA for SMC frame is disabled. The six dedicated 1-1 slot corresponding SMC registers can only be read by the CPU through the SFR. Reading the DSFRs returns 0. 1 DMA for SMC frame is enabled. The six dedicated slot corresponding SMC registers can be read by reading the DSFR successively. In the DMA mode the six registers are read in a round robin fashion as explained in the DSFR details. Reading directly through the SFR registers is still available.
30–31 DMA_PM_DS_CONFIG	These bits define how the PSI5 messages are stored in the MEM_DEPTH memory area and the associated diagnostic bits (EISR, NDSR) are transferred. The DPMR, DDSR, and DSFR registers should be used for DMA access and the PMRH/PMRL, NDSR, EISR, and OWSR registers should be used for CPU access. NOTE: For more details please see the description of the DCR[DMA_PM_DS_WM] bits. When DMA_PM_DS_CONFIG bits = conf2 or conf3 and if the CPU and the DMA are simultaneously used for reading the PSI5 messages, then in the course of reading the messages it is possible that there can be an overflow, though some intermediate locations, which have been read by the CPU, are empty. This will occur because the DMA access is made independent of the CPU access. The DMA logic would never know that the CPU has already read a message which was scheduled to be read by the DMA. Whenever the DMA is used for reading the PSI5 message (conf2/conf3) modes then though the CPU can read the messages in parallel through the PMRH/PMRL registers, still for the purpose of setting the Overwrite/Overflow bits only a read from the DMA through the popup registers (DPMR) would be treated as valid read. This will also be true for the SMC messages when accessed simultaneously by the CPU/DMA. NOTE: These bits are writable only in config mode. 00 conf1 The DMA request is disabled. In this mode, only the interrupts can be used for indicating that data transfer is required. 01 conf2 After transferring the "dma_pm_ds_wm" number of PSI5 messages, there will be two additional 32-bit transfers in which the NDSR and the EISR registers are sequentially transferred through the DPMR registers by using DMA. 10 conf3 "dma_pm_ds_wm" number of PSI5 messages is transferred in each DMA request. The DMA read is through the DPMR registers. 11 conf4

Table continues on the next page...

PSI5_CH0_DCR field descriptions (continued)

Field	Description
	Only the diagnostic bits are transferred through the DDSR. The DMA read is through the DDSR registers.

47.4.4 DMA Status Register (PSI5_CH0_DSR)

This section defines the DMA Status Register.

Address: 0h base + 10h offset = 10h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0												IS_DMA_TF_SF	IS_DMA_TF_PM_DS	0	
W													w1c	w1c		
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0				IS_DMA_PM_DS_FIFO_FULL	IS_DMA_SFUF	0	IS_DMA_PM_DS_UF	0							
W					w1c	w1c		w1c								
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

PSI5_CH0_DSR field descriptions

Field	Description
0–12 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
13 IS_DMA_TF_SF	This flag is set when DMA transfer finishes for DMA SMC Frame register.
14 IS_DMA_TF_PM_DS	This flag is set when DMA transfer finishes . For the various configurations of this request please refer to the descriptions of the DMA_PM_DS_CONFIG bits in the DCR. This flag is cleared by w1c.
15–19 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
20 IS_DMA_PM_DS_FIFO_FULL	Interrupt Status when there is FIFO FULL corresponding to DMA request. The FIFO FULL scenarios are detailed below with respect to the various settings of the DMA_PM_DS_CONFIG bit. conf1: FIFO FULL disabled conf2 and conf3: FIFO FULL occurs when all the memory locations become full in the PSI5 message FIFO. It indicates that without any read, any new upcoming message will now overwrite the existing messages and set the overwrite bits. This bit is cleared either by w1c or upon a PSI5 message read depending on the FAST_CLR_PSI5 bit. conf4: FIFO FULL occurs when there is a "REGISTER FULL" in the Diagnostic registers. FIFO FULL here means that the diagnostic registers contain 32 unread diagnostic bits. A new message starts changing the status of the diagnostic bits. In this configuration it is necessary to read the diagnostic bits through the DMA using the DDSR in order to prevent FIFO FULL. Of course the diagnostic registers NDSR/EISR can be read any time by the CPU, but to prevent FIFO FULL they HAVE to be read through the DDSR registers only. Also in this mode the PSI5 messages DONT contribute to the FIFO FULL. The PSI5 messages have to be read by the CPU using the PMRH/PMRL registers in order to prevent the overwrite bits from being set in the OWSR. In this mode the OWSR bits are NOT automatically cleared upon the PSI5 message read since the PSI5 message has to be read by the PMRH/PMRL register and NOT though the DPMR. The OWSR bits have to be cleared by a w1c. Reading the DPMR returns a 0 in this mode. This bit is cleared by a w1c. 0 No FIFO full. 1 FIFO full has occurred
21 IS_DMA_SFUF	SMC Frame DMA underflow: This happens when the DSFR has been read without a proper DMA request being asserted. The DSFR is empty and it is read. This bit is cleared by a w1c. 0 No underflow has occurred. 1 Underflow has occurred.
22 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
23 IS_DMA_PM_DS_UF	Depending on the DMA_PM_DS_CONFIG bits following is the underflow conditions: Default value is 0. conf1: underflow disabled conf2 and conf3: Underflow happens when the software reads the PSI5 message FIFO though the DPMR, beyond the available messages (empty area). conf4: Underflow happens when the diagnostic registers are read through the DDSR without a valid DMA request and the DDSR contains no new data. The bits are cleared by a w1c.

Table continues on the next page...

PSI5_CH0_DSR field descriptions (continued)

Field	Description
0	No underflow has occurred.
1	Underflow has occurred.
24–31 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

47.4.5 General Interrupt Control Register (PSI5_CH0_GICR)

The GICR contains the bits for controlling interrupts related to:

- ECU-to-sensor communication
- the Time Stamp

Address: 0h base + 14h offset = 14h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0					0			IE_ST	IE_DTS	IE_DSROW	IE_BROW	IE_PROW	IE_DSRR	IE_BRR	IE_PRR
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0					0			0				0			
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

PSI5_CH0_GICR field descriptions

Field	Description
0–1 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
2–7 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
8 IE_ST	Interrupt request enabled when the Sync Pulse Triggered Time Stamp value is refreshed on STSRR. NOTE: This bit is writable only in Config and Normal modes. 0 Interrupt is disabled. 1 Interrupt is enabled.
9 IE_DTS	Interrupt request enabled when the Data Start sequence Triggered Time Stamp value is refreshed on DTSRR. NOTE: This bit is writable only in Config and Normal modes.

Table continues on the next page...

PSI5_CH0_GICR field descriptions (continued)

Field	Description
	0 Interrupt is disabled. 1 Interrupt is enabled.
10 IE_DSROW	Interrupt request enabled when system tries to overwrite on Data Shift register when it is not ready to accept new data. NOTE: This bit is writable only in Config and Normal modes. 0 Interrupt is disabled. 1 Interrupt is enabled.
11 IE_BROW	Interrupt request enabled when system tries to overwrite on buffer register when it is not ready to accept new data. NOTE: This bit is writable only in Config and Normal modes. 0 Interrupt is disabled. 1 Interrupt is enabled.
12 IE_PROW	Interrupt request enabled when system tries to overwrite on Preparation register when it is not ready to accept new data. NOTE: This bit is writable only in Config and Normal modes. 0 Interrupt is disabled. 1 Interrupt is enabled.
13 IE_DSRR	Interrupt request enabled when Data Shift Register is ready to accept new data. NOTE: This bit is writable only in Config and Normal modes. 0 Interrupt is disabled. 1 Interrupt is enabled.
14 IE_BRR	Interrupt request enabled when buffer Register is ready to accept new data. NOTE: This bit is writable only in Config and Normal modes. 0 Interrupt is disabled. 1 Interrupt is enabled.
15 IE_PRR	Interrupt request enabled when Preparation Register ready to accept new data. NOTE: This bit is writable only in Config and Normal modes. 0 Interrupt is disabled. 1 Interrupt is enabled.
16–17 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
18–23 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
24–25 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
26–31 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

47.4.6 New Data Interrupt Control Register (PSI5_CH0_NDICR)

The NDICR contains the interrupt enable bits related to a new PSI5 message arrival, in the corresponding PSI5 message buffer location.

NOTE

Depending on the value of MEM_DEPTH in the PCCR, only the corresponding diagnostic bits are set in this register.

Address: 0h base + 18h offset = 18h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	IE_ND																															
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	

PSI5_CH0_NDICR field descriptions

Field	Description
0–31 IE_ND	Interrupt request enabled when any new message (fault-free/with fault) is received in the RAM buffer Register 'x' location [x: 0 to 31], This interrupt is generated only when corresponding RAM buffer Ready with new data. NOTE: These bits are writable only in Config and Normal modes. 0 Interrupt is disabled. 1 Interrupt is enabled.

47.4.7 Overwrite Interrupt Control Register (PSI5_CH0_OWICR)

The Overwrite Interrupt Control Register (PSI5_OWICR) is related to the generation of the overwrite interrupts. The overwrite interrupt is generated when the unread PSI5 message in a particular RAM location is overwritten by another PSI5 message.

NOTE

Depending on the value of MEM_DEPTH in the PCCR, only the corresponding diagnostic bits are set in this register.

Address: 0h base + 1Ch offset = 1Ch

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	IE_OW																															
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	

PSI5_CH0_OWICR field descriptions

Field	Description
0–31 IE_OW	Interrupt request enabled when any new message overwrites the old unread PSI5 message in the RAM buffer Register 'x' location [x: 0 to 31], This interrupt generated only when corresponding RAM buffer Ready with new data. NOTE: These bits are writable only in Config and Normal modes. 0 Interrupt is disabled. 1 Interrupt is enabled.

47.4.8 Error Interrupt Control Register (PSI5_CH0_EICR)

The following figure shows the Error Interrupt Control Register.

NOTE

Depending on the value of MEM_DEPTH in the PCCR, only the corresponding diagnostic bits are set in this register.

Address: 0h base + 20h offset = 20h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R																																
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

PSI5_CH0_EICR field descriptions

Field	Description
0–31 IE_ERROR	Interrupt request enabled when any/all of the error conditions C, E, EM, T, or F is observed in a PSI5 message in the RAM buffer Register 'x' location [x: 0 to 31], This interrupt is generated only when corresponding RAM buffer Ready with new data. Which of the error condition (any of C, E, EM, T, or F) would generate the interrupt is selectable by PCCR[ERROR_SELECT] field. If all are selected then which out of C,E,EM,T,F is set can be isolated by reading corresponding received message. NOTE: These bits are writable only in Config and Normal modes. 0 Interrupt is disabled. 1 Interrupt is enabled.

47.4.9 General Interrupt Status Register (PSI5_CH0_GISR)

The GISR contains the status bits related to SMC messages, ECU-to-sensor communication, and the Time Stamp.

Address: 0h base + 24h offset = 24h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	IS_DB_FR	0	IS_CESM						IS_STS	IS_DTS	IS_DSROW	IS_BROW	IS_PROW	DSR_RDY	DBR_RDY	DPR_RDY
W			w1c						w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	IS_OWSM							0	IS_NVSM						
W		w1c								w1c						
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

PSI5_CH0_GISR field descriptions

Field	Description
0 IS_DB_FR	This flag is set to “1” when the IP enters the Debug freeze mode. Please see Debug mode for details about the Debug mode. This bit is auto cleared by the hardware when the IP exits the debug freeze mode. 0 IP not in debug freeze mode. 1 IP in debug freeze mode.
1 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
2–7 IS_CESM	These flags are set when the received SMC frame in the corresponding slot has a CRC failure (CRC recalculation on SMC). These bits are automatically cleared upon message read, when the FAST_CLR_SMC = 1 and the DMA_EN_SF = 1.
8 IS_STS	This Interrupt flag is set when the Sync Pulse Triggered Time Stamp value is refreshed on STSRR.

Table continues on the next page...

PSI5_CH0_GISR field descriptions (continued)

Field	Description
9 IS_DTS	This Interrupt flag is set when the Data Start sequence Triggered Time Stamp value is refreshed on DTSRR.
10 IS_DSROW	This flag is set when the system tries to overwrite on Data Shift register when it is not ready to accept new data.
11 IS_BROW	This flag is set when the system tries to overwrite on buffer register when it is not ready to accept new data.
12 IS_PROW	This flag is set when the system tries to overwrite on Preparation register when it is not ready to accept new data.
13 DSR_RDY	This bit acts both as a status and a control bit. Status bit Action : When "1" it indicates that the Data Shift register is ready to receive new data from the system bus side or the Data Buffer register. When "0" it indicates that there is a ongoing command transmission of the data in DSR , or the CPU has done a "w1c" to this bit leading to the new command transmission . In this case any write by the CPU to this register will be rejected and the PSI5_GISR[IS_DSROW] bit gets set. A write to PSI5_DOBCR[DSR_RST] will abort the current command transmission and make the DSR_RDY as "1". Control Bit Action : When a "w1c" is done to this register bit, and PSI5_DOBCR[SW_READY] == 1 , then it indicates to the IP that the data in DSR is valid and is to be used for command shifting . This procedure is used when the data is written by the CPU to DSR directly skipping the DBR. For detailed action of this bit please refer to Data transmission . NOTE: For this field, Individual "bit setting" commands should not be used. Rather, the normal register read/write commands should be used. NOTE: In configuration mode, writes to this bit are ignored.
14 DBR_RDY	This bit acts both as a status and a control bit. Status bit Action : When "1" it indicates that the Data Buffer register is ready to receive new data from the system bus side or the Data Preparation register. When "0" it indicates that there is a ongoing shift of the data from the DPR to the DBR, or the CPU has done a "w1c" to this bit . In this case any write by the CPU to this register will be rejected and the PSI5_GISR[IS_BROW] bit gets set. A write to PSI5_DOBCR[DBR_RST] will abort the current data transaction in DBR and make the DBR_RDY as "1". Control Bit Action : When a "w1c" is done to this register bit, then it indicates to the IP that the data in DBR is valid and is to be used for shifting to DSR . This procedure is used when the data is written by the CPU to DBR directly skipping the DPR. For detailed action of this bit please refer to Data transmission . NOTE: For this field, individual "bit setting" commands should not be used. Rather, the normal register read/write commands should be used. NOTE: In configuration mode, writes to this bit are ignored.
15 DPR_RDY	This bit acts both as a status and a control bit. Status bit Action : When "1" it indicates that the Data Preparation register is ready to receive new data from the system bus side . When "0" it indicates that there is a ongoing processing of the data in DPR or there is a data shift ongoing from the DPR to the DBR, or the CPU has done a "w1c" to this bit . In this case any write by the CPU to this register will be rejected and the PSI5_GISR[IS_PROW] bit gets set. When PSI5_DOBCR[CMD_TYPE] == "7" then this bit remains at value "0". Control Bit Action : When a "w1c" is done to this register bit, then it indicates to the IP that the data in DPR is valid and is to be used for processing and shifting to DBR . For detailed action of this bit please refer to Data transmission .

Table continues on the next page...

PSI5_CH0_GISR field descriptions (continued)

Field	Description
	NOTE: For these register bits individual "bit setting" commands should not be used. Rather, the normal register read/write commands should be used. NOTE: In configuration mode, writes to this bit are ignored.
16–17 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
18–23 IS_OWSM	Status for the SMC message overwrite bits. Each of the 6 locations correspond to the SMC message pertaining to each SFR[x] register, x = 1 to 6. These bits are automatically cleared upon message read when the FAST_CLR_SMC = 1 and the DMA_EN_SF = 1.
24–25 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
26–31 IS_NVSM	These flags are set when corresponding SFR[i] receives and is ready (to read) with new valid SMC message. These bits are automatically cleared upon message read, when the FAST_CLR_SMC = 1 and the DMA_EN_SF = 1.

47.4.10 DMA PSI5 Message Register (PSI5_CH0_DPMR)

This section defines the DMA PSI5 Message Register.

Address: 0h base + 28h offset = 28h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	PSI5_RXDATA																															
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	

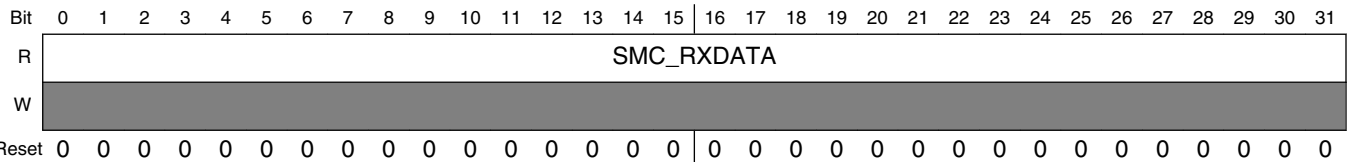
PSI5_CH0_DPMR field descriptions

Field	Description
0–31 PSI5_RXDATA	PSI5_RXDATA[31:0] This popup register contain the PMRL[31:0] data followed by the PMRH[31:0] data for each of the FIFO locations, when the FIFO is read by the DMA. It would require two 32-bit IPS access to read one PSI5 Message. For each of the locations the DATA is pushed sequentially till all the data corresponding to the programmed watermark is complete after which the DMA request goes down. The depth of the FIFO is 32x64. This popup register can be used differently based on the configuration of the DMA_PM_DS_CONFIG bits in the DCR. When DMA_PM_DS_CONFIG=conf2 then it pops the PSI5 message followed by NDSR followed by EISR. When DMA_PM_DS_CONFIG= conf3, it pops only the PSI5 messages. When DMA_PM_DS_CONFIG=conf4 then it pops the NDSR followed by EISR. This register is a reflection of the location, pointed to by the DMA read pointer. For more details please refer to the PSI5 Channel Control Register (PSI5_CH0_PCCR) , DMA Control Register (PSI5_CH0_DCR) and DMA Status Register (PSI5_CH0_DSR) .

47.4.11 DMA SMC Frame Register (PSI5_CH0_DSFR)

This section shows the DMA SMC Frame Register.

Address: 0h base + 2Ch offset = 2Ch



PSI5_CH0_DSFR field descriptions

Field	Description
0–31 SMC_RXDATA	When the DMA_EN_SF = 1 then the six SFR registers are searched in a round robin fashion for the reception of the complete SMC data. The DMA request is asserted as soon as the first encountered SFR has a complete SMC frame. This request remains asserted till all the registers which have a complete data ready have transferred the data through the DMA. Each other SFR is a 32-bit register. This register is a reflection of the location, pointed to by the DMA read pointer.

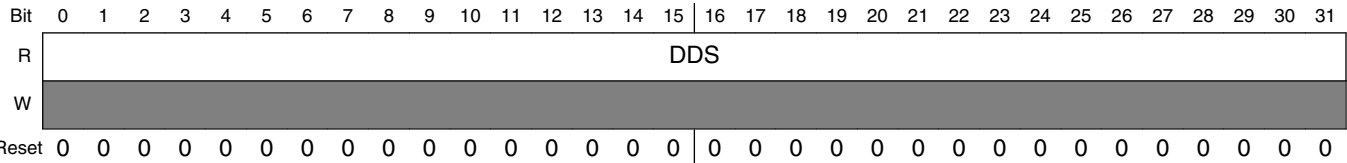
47.4.12 DMA Diagnostic Status Register (PSI5_CH0_DDSDR)

This section defines the DMA Diagnostic Status Register.

NOTE

No FIFO is implemented for diagnostic registers. They are two 32-bit registers addressable by either the CPU (reading NDSR or EISR directly) or the DMA (reading the DDSR successively).

Address: 0h base + 30h offset = 30h



PSI5_CH0_DDSDR field descriptions

Field	Description
0–31 DDS	This register maps each of the individual Diagnostic registers (in the order NDSR, EISR) depending on the configuration of the DMA_PM_DS_CONFIG bits in the DCR. This register is a reflection of the location, pointed to by the DMA read pointer.

PSI5_CH0_DDSR field descriptions (continued)

Field	Description
	For more details please refer to the PSI5 Channel Control Register (PSI5_CH0_PCCR) , DMA Control Register (PSI5_CH0_DCR) and DMA Status Register (PSI5_CH0_DSR) .

47.4.13 PSI5 Message Receive Register Low (PSI5_CH0_PMRRL)

This section defines the PSI5 Message Receive Register Low.

Address: 0h base + 34h offset = 34h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	DATA_REGION															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	DATA_REGION												CRC			C
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

PSI5_CH0_PMRRL field descriptions

Field	Description
0–27 DATA_REGION	These are application-specific optional bits + data payload of received message. See Reception of data frames for more details. The data is left-aligned i.e. Data Region[0]/D0 corresponds to first received bit of the PSI5 message after Start sequence detection and Data Region[n]/D'n' corresponds to successive bits up to expected number of bits (n: 8 ? n ? 28). If n < 28 remaining bit positions are filled with '0's i.e. between last received bit and CRC field.
28–30 CRC	This field represents CRC/Parity value of the data. When parity configuration is selected in the S[1-6]FCR registers, the CRC[2] bits denotes the parity.
31 C	This bit will be set if CRC/P recalculation return an Error.

47.4.14 PSI5 Message Receive Register High (PSI5_CH0_PMRRH)

This section defines the PSI5 Message Receive Register High.

NOTE

Any new PSI5 message is always stored in the PSI5 receive register (PMRRL/PMRRH). In parallel it is also stored in the RAM registers in a sequential manner like in a Ring Buffer. When a new message comes it would always overwrite the data in the PSI5 receive register. But the original message would still be available in the RAM register provided all locations in the RAM registers are not full, and an overwrite has not occurred.

Address: 0h base + 38h offset = 38h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	F	EM	E	T	SlotCounter			TimeStampValue							
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	TimeStampValue															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

PSI5_CH0_PMRRH field descriptions

Field	Description
0 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
1 F	This represents that NO frame was received in the corresponding configured slot. 0 Frame was received in the corresponding configured slot. 1 No frame was received (F bit error).
2 EM	This bit indicates electrical error in the PSI5 Message at bits corresponding to Serial Messaging Channel (M0,M1), that is, at least one bit is absent during the T bit period. 0 No electrical error in M0, M1 fields. 1 Electrical error in M0, M1 fields.
3 E	This bit indicates electrical error, i.e. at least one bit is absent during the Tbit period (except M0 and M1 bits and the start bits). 0 No electrical error in fields other than M0, M1, and start fields. 1 Electrical error in fields other than M0, M1, and start fields.
4 T	This bit indicates timing error i.e. frame has started in an unconfigured slot, spread across two slots, started before the first configured slot. 0 No timing error in the corresponding slot. 1 Timing error in the corresponding slot.
5–7 SlotCounter	This value indicates the slot number in which this frame was received. More specifically, it indicates in which slot the start bits of the frame have been detected. If the start bits get detected before the start

Table continues on the next page...

PSI5_CH0_PMRRH field descriptions (continued)

Field	Description
	boundary of Slot1, then the slot counter has a value of "0". For asynchronous modes the slot counter has a fixed value of "1". The six slots are numbered as Slot1, Slot2,...Slot6.
8–31 TimeStampValue	This is 24-bit Time Stamp value appended to Received message as soon as start bits are detected. This time stamp is captured at rising edge of S0 and stored at the rising edge of S1. The specific value that is appended can be programmed by setting the TS_CAPT bit in the Slot n Frame Configuration Register (PSI5_CH0_SnFCR) . In other words these bits contain either the start bit(S0) captured time stamp or the SYNC pulse captured time stamp, dependant on TS_CAPT bit. Note that in messages where "F" bit has been set ; the value of the Time Stamp appended to the message is "0" . It is possible to reset the time stamp counter by asserting to "1" the "gtm_reset" signal from the GTM.

47.4.15 PSI5 Message Register Low i (PSI5_CH0_PMRLn)

The following figure shows the PSI5 Message Register Low i.

Address: 0h base + 3Ch offset + (8d × i), where i=0d to 31d

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	DATA_REGION															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	DATA_REGION												CRCP		C	
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

PSI5_CH0_PMRLn field descriptions

Field	Description
0–27 DATA_REGION	These are application-specific optional bits + data payload of received message. See Reception of data frames for more details. The data is left-aligned i.e. Data Region[0]/D0 corresponds to first received bit of the PSI5 message after Start sequence detection and Data Region[n]/D'n' corresponds to successive bits up to expected number of bits (n: 8 ? n ? 28). If n < 28 remaining bit positions are filled with '0's i.e. between last received bit and CRC field. This bit is writeable in Config mode.
28–30 CRCP	This field represents CRC/Parity value of the data. When parity configuration is selected in the S[1-6]FCR registers, the CRC[2] bits denotes the parity. This bit is writeable in Config mode.
31 C	This bit will be set if CRC/P recalculation return an Error. This bit is writeable in Config mode.

47.4.16 PSI5 Message Register High i (PSI5_CH0_PMRHn)

The following figure shows the PSI5 Message Register High i.

Address: 0h base + 40h offset + (8d × i), where i=0d to 31d

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	O								TimeStampValue							
W		F	EM	E	T	Slot_Counter										
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	TimeStampValue															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

PSI5_CH0_PMRHn field descriptions

Field	Description
0 O	This bit carries the same information as OWSR, but appended here for each message separately. This bit is set when the current message (which is being read) has overwritten some previous unread message. Note that, if the corresponding location in the OWSR has been cleared or has been set by the SOWSR, then the same will be reflected in the message stored in the corresponding RAM location as well.
1 F	This represents that NO frame was received in corresponding configured Slot. This bit is writable in Config mode (for Test purposes). 0 Frame was received in the corresponding configured slot. 1 No frame was received (F bit error).
2 EM	This bit indicates electrical error in the PSI5 Message at bits corresponding to Serial Messaging Channel(M0, M1) i.e. at least one bit is absent during the Tbit period. This bit is writable in Config mode (for Test purposes). 0 No electrical error in M0, M1 fields. 1 Electrical error in M0, M1 fields.
3 E	This Bit Indicates Electrical Error i.e. at least one bit is absent during the Tbit period (except M0 and M1 bits and the start bits). This bit is writable in Config mode (for Test purposes). 0 No electrical error in fields other than M0, M1 and start fields. 1 Electrical error in fields other than M0, M1 and start fields.
4 T	This bit indicates timing error i.e. frame has started in an unconfigured slot or it has spread across two slots or it has started before the first configured slot. This bit is writable in Config mode (for Test purposes).

Table continues on the next page...

PSI5_CH0_PMRHn field descriptions (continued)

Field	Description
	0 No timing error in the corresponding slot. 1 Timing error in the corresponding slot.
5–7 Slot_Counter	This value indicates the slot number in which this frame was received. More specifically, it indicates in which slot the start bits of the frame have been detected. If the start bits get detected before the start boundary of Slot1, then the slot counter has a value of "0". The slot counter is automatically incremented at each slot boundary irrespective of whether a message is received or not. For asynchronous modes the slot counter has a fixed value of "1". The six slots are numbered as Slot1, Slot2,...Slot6. This bit is writable in Config mode (for Test purposes).
8–31 TimeStampValue	This is 24-bit Time Stamp value appended to Received message as soon as start bits are detected. This time stamp is captured at rising edge of S0 and stored at the rising edge of S1. The specific value that is appended can be programmed by setting the TS_CAPT bit in the Slot n Frame Configuration Register (PSI5_CH0_SnFCR) . In other words these bits contain either the start bit (S0) captured time stamp or the SYNC pulse captured time stamp, dependant on TS_CAPT bit. Note that in messages where "F" bit has been set ; the value of the Time Stamp appended to the message is "0". It is possible to reset the time stamp counter by asserting to "1" the "gtm_reset" signal from the GTM. This bit is writable in Config mode (for Test purposes).

47.4.17 SMC Frame Register n (PSI5_CH0_SFRn)

The following figure shows the SMC Frame Register n.

NOTE

Six dedicated 32-bit SFR[x] registers for SMC are available with a 1-1 slot correspondence. These can be read by either the CPU or the DMA (when DMA_EN_SF = 1).

NOTE

No FIFO for SMC frame is available. Six 32-bit registers are available in the address space, which can be accessed by either the DMA (using the DSFR) or the CPU (using the six SFR registers), depending on the setting of the DMA_EN_SMC bit.

Address: 0h base + 13Ch offset + (4d × i), where i=0d to 5d

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	SLOT_NO			CER	OW	CRC					C	ID				
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	IDDATA				DATA											
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

PSI5_CH0_SFRn field descriptions

Field	Description
0–2 SLOT_NO	This indicates in which slot this SMC frame was received. Note that here the slot number refers to the slot in which all the re-arranged PSI5 messages (from which the M0,M1 bits have been extracted) are supposed to have occurred after the correct PSI5 message placement and Timing corrections have been performed. For SFR[1], slot_no = 1; For SFR[2], slot_no = 2; For SFR[3], slot_no = 3; For SFR[4], slot_no = 4; For SFR[5], slot_no = 5; For SFR[6], slot_no = 6; This field is writable in Config mode (for Test purposes).
3 CER	CRC Error This field is writable in Config mode (for Test purposes). 0 The present data has no CRC error. 1 The present data has a CRC error.
4 OW	Overwrite status NOTE: If the corresponding SOWSM bits in the SSESr have been set or the OWSM bits in the GISR have been cleared, then the same will be reflected in the message stored in the corresponding RAM location as well. The overwrite occurs when any unread SMC message in a particular SMC slot register gets overwritten by a new SMC message of that particular slot. Even if the SMC message has a CRC error, still if it is unread and is overwritten, the overwrite bit gets set. 0 The present data was already read when new message arrived. 1 New message has overwritten the previous unread message.
5–10 CRC	6-bit CRC for slow serial message This field is writable in Config mode (for Test purposes).
11 C	Configuration bit This bit is writable in Config mode (for Test purposes). 0 12-bit data and 8-bit message ID. 1 16-bit data and 4-bit message ID.
12–15 ID	Message ID: If C = '0' indicates ID[7:4], if C = '1' indicates ID[3:0]. This field is writable in Config mode (for Test purposes).
16–19 IDDATA	Message ID/DATA: If C = '0' indicates ID[3:0], if C = '1' indicates DATA[15:12]. This field is writable in Config mode (for Test purposes).
20–31 DATA	DATA payload This field is writable in Config mode (for Test purposes).

47.4.18 New Data Status Register (PSI5_CH0_NDSR)

The following figure shows the New Data Status Register.

Address: 0h base + 154h offset = 154h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R																NDS																
W																w1c																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	

PSI5_CH0_NDSR field descriptions

Field	Description
0–31 NDS	New Data Status flags for PSI5 Messages corresponding to each PSI5 MB locations. These bits are set when a new Message arrives i.e. faulty or non faulty in corresponding PSI5 MB location. Bit 0 reflects the MB0 location. These bits can be cleared by w1c or the fast clearing. Please refer to PCCR, DCR and DSR registers for more information. Also, these bits can be set independent by writing in SNDR.

47.4.19 Overwrite Status Register (PSI5_CH0_OWSR)

Address: 0h base + 158h offset = 158h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R																OWS																
W																w1c																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	

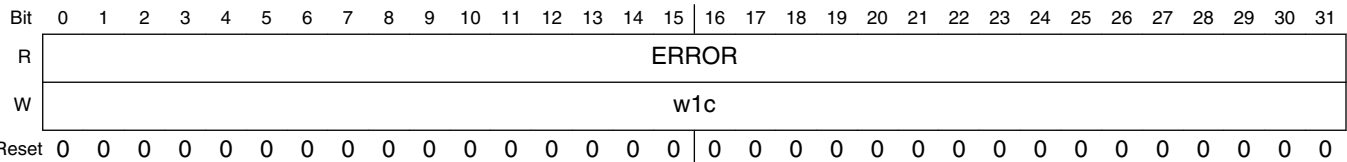
PSI5_CH0_OWSR field descriptions

Field	Description
0–31 OWS	Over Write Status flags for PSI5 Messages corresponding to each PSI5 location in the RAM registers. These bits are set when any unread receive PSI5 message is overwritten by a newly arrived message). These bits can be cleared by w1c or the fast clearing. Please refer to PCCR, DCR and DSR registers for more information. Also, these bits can be set independent by writing in SOWSR.

47.4.20 Error Indication Status Register (PSI5_CH0_EISR)

The following figure shows the Error Indication Status Register.

Address: 0h base + 15Ch offset = 15Ch



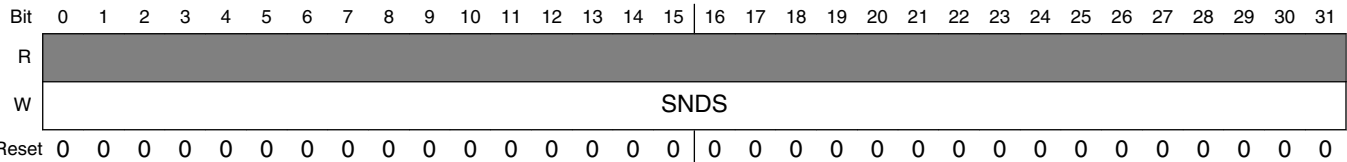
PSI5_CH0_EISR field descriptions

Field	Description
0–31 ERROR	Error Status flags for PSI5 Messages corresponding to each PSI5 MB locations. These bits are set when any of the five selectable (by ERROR_SELECT) bits (C, E, EM, T or F) is set for the corresponding PSI5 Message. These bits can be cleared by w1c or the fast clearing. Please refer to PCCR, DCR and DSR registers for more information. Also, these bits can be set independently by writing in SEISR.

47.4.21 Set New Data Status Register (PSI5_CH0_SNDSR)

The following figure shows the Set New Data Status Register.

Address: 0h base + 160h offset = 160h



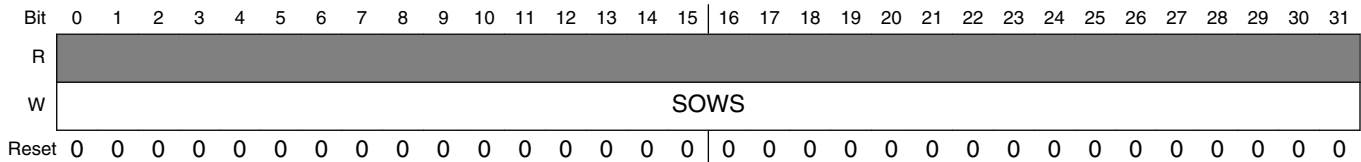
PSI5_CH0_SNDSR field descriptions

Field	Description
0–31 SNDS	Sets New Data Status flags for PSI5 Messages corresponding to each PSI5 MB locations. The corresponding NDS Flags can be set by writing to corresponding bits in this registers. This register is required for IP testing purposes (testing of Error Conditions through software). These bits can SET the status flags in parallel to the hardware events and are used for test purpose. These registers are one-shot and would always be read as '0'. NOTE: This bit is writable only in config and Normal modes.

47.4.22 Set Overwrite Status Register (PSI5_CH0_SOWSR)

The following figure shows the Set Overwrite Status Register.

Address: 0h base + 164h offset = 164h



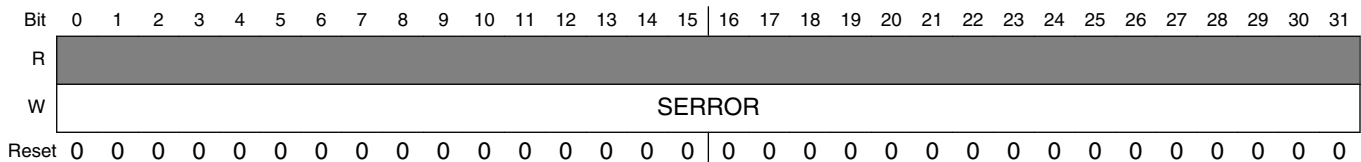
PSI5_CH0_SOWSR field descriptions

Field	Description
0–31 SOWS	Sets overwrite status flags for PSI5 messages corresponding to each PSI5 MB locations. The corresponding OWS flags can be set by writing to corresponding bits in this registers. This register is required for IP testing purposes (testing of Error Conditions through software). These bits can set the status flags in parallel to the hardware events and are used for test purpose. These registers are one-shot and would always be read as 0. NOTE: These bits are writable only in Config and Normal modes.

47.4.23 Set Error Status Register (PSI5_CH0_SEISR)

The following figure shows the Set Error Status Register.

Address: 0h base + 168h offset = 168h



PSI5_CH0_SEISR field descriptions

Field	Description
0–31 SERROR	Sets Error Status flags for PSI5 messages corresponding to each PSI5 MB locations. The corresponding ERROR flags can be set by writing to corresponding bits in this registers. This register is required for IP testing purposes (testing of Error Conditions through software). These bits can set the status flags in parallel to the hardware events and are used for test purpose. These registers are one-shot and would always be read as 0. NOTE: These bits are writable only in Config and Normal modes.

47.4.24 Set SMC Error Status Register (PSI5_CH0_SSESR)

The following figure shows the Set SMC Error Status Register.

Address: 0h base + 16Ch offset = 16Ch

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0						0									0						0	0								
W	SCESM																		SOWSM								SNVSM					
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

PSI5_CH0_SSESR field descriptions

Field	Description
0–1 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
2–7 SCESM	Sets IS_CESM Status flags for SMC Messages corresponding to each SMC MB locations. The corresponding IS_CESM Flags can be set by writing to corresponding bits in this registers. This register is required for IP testing purposes (testing of Error Conditions through software). These bits can SET the status flags in parallel to the hardware events and are used for test purpose. These registers are one-shot and would always be read as '0'. NOTE: These bits are writable only in Config and Normal modes.
8–17 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
18–23 SOWSM	Sets IS_OWSM Status flags for SMC Messages corresponding to each SMC MB locations. The corresponding IS_OWSM Flags can be set by writing to corresponding bits in this registers. This register is required for IP testing purposes (testing of Error Conditions through software). These bits can SET the status flags in parallel to the hardware events and are used for test purpose. These registers are one-shot and would always be read as '0'. NOTE: These bits are writable only in Config and Normal modes.
24–25 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
26–31 SNVSM	Sets IS_NVSM Status flags for SMC Messages corresponding to each SMC MB locations. The corresponding IS_NVSM Flags can be set by writing to corresponding bits in this registers. This register is required for IP testing purposes (testing of Error Conditions through software). These bits can SET the status flags in parallel to the hardware events and are used for test purpose. These registers are one-shot and would always be read as '0'. NOTE: This bit is writable only in Config and Normal modes.

47.4.25 Sync Time Stamp Read Register (PSI5_CH0_STSR)

The following figure shows the Sync Time Stamp Read Register.

Address: 0h base + 170h offset = 170h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0								STSV																							
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	

PSI5_CH0_STSR field descriptions

Field	Description
0–7 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
8–31 STSV	Sync Time Stamp Value[23:0] This bitfield provides the value of the free-running 24-bit time stamp counter triggered (captured) at last sync pulse.

47.4.26 Data Time Stamp Read Register (PSI5_CH0_DTSR)

The following figure shows the Data Time Stamp Read Register.

NOTE

If S0 and S1 are detected in different slots then the Time Stamp value and the Slot Counter, both will correspond to the first slot(S0)

Address: 0h base + 174h offset = 174h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0					SLOT_COUNTER			DTSV							
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	DTSV															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

PSI5_CH0_DTSRR field descriptions

Field	Description
0–4 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
5–7 SLOT_ COUNTER	3bit Slot Counter which corresponds to the slot for which the DTSRR contains the Time Stamp. The slot_counter contains the slot in which the start bits of this frame was captured (after successful detection of S1). If the start bits get detected before the start boundary of Slot1, then the slot counter has a value of "0". The slot counter is automatically incremented at each slot boundary irrespective of whether a message is received or not. In case of asynchronous mode the slot counter has a fixed value of "1".
8–31 DTSV	Data Time Stamp Value[23:0] Whenever a valid sequence of the start bits (S0,S1) is detected then the value of the Time Stamp counter captured at the rising edge of the S0, is stored in this register at the rising edge of S1.

47.4.27 Slot n Frame Configuration Register (PSI5_CH0_SnFCR)

The following figure shows the Slot n Frame Configuration Register.

NOTE

In Asynchronous mode, the PSI5_S1FCR register has to be programmed for controlling the various parameters pertaining to the message received. The other registers PSI5_S[2-6]FCR have no effect in the asynchronous mode.

Address: 0h base + 178h offset + (4d × i), where i=0d to 5d

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0												SLOT_EN	TS_CAPT	0	
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31				
R	SMCL	0										DRL								CRCP
W																				
Reset	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0				

PSI5_CH0_SnFCR field descriptions

Field	Description
0–12 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

Table continues on the next page...

PSI5_CH0_SnFCR field descriptions (continued)

Field	Description
13 SLOT_EN	SLOT_EN 0 This particular slot is disabled i.e not in use and corresponding configuration bits (as defined by other bits of this PSI5_SnFCR) have no effect. 1 This particular slot is enabled i.e is in use and corresponding configuration bits (as defined by other bits of this PSI5_SnFCR) are applicable.
14 TS_CAPT	TS_CAPT 0 Time stamp value captured at rising edge of S0 and stored Start sequence first edge. 1 Time stamp value captured at corresponding Tsync pulse (posedge) and stored with corresponding slot PSI5 message.
15 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
16 SMCL	Serial Messaging Channel field This field is used to enable the detection of start sequence of SMC frame on particular slot. Detection is not done if this field is '0'. 0 Serial Messaging Channel (optional) not present on Rx Message during Time Slot 'n' 1 2-bit{M1,M0} Serial Messaging Channel present on Rx Message during Time Slot 'n'
17–25 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
26–30 DRL	Data Region Length[4:0] This field represents Length of Data region (including Frame Control, Status , Data payload1&2 regions and optional messaging bits M0,M1) in Rx Message during Time Slot 'n' 00000-00111: Invalid 01000: 8 bit long data region 01001: 9 bit long data region 01010: 10 bit long data region 11011: 27 bit long data region 11100: 28 bit long data region 11101-11111: Invalid
31 CRCP	CRCP 0 3 bit CRC[2:0] present on Rx Message during Time Slot 'n' 1 1 bit Parity field present on Rx Message during Time Slot 'n'

47.4.28 Slot 1 Start Boundary Register (PSI5_CH0_S1SBR)

The following figure shows the Slot 1 Start Boundary Register.

Address: 0h base + 190h offset = 190h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Read	0	S1SBT														
Write																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

PSI5_CH0_S1SBR field descriptions

Field	Description
0 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
1–15 S1SBT	Slot 1 Start Boundary Time This field specifies the time in intervals of 1 μs, occurring after the rising edge of the timing sync pulse, at which the slot 1 should start.

47.4.29 Slot 2 Start Boundary Register (PSI5_CH0_S2SBR)

The following figure shows the Slot 2 Start Boundary Register.

Address: 0h base + 192h offset = 192h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Read	0	S2SBT														
Write																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

PSI5_CH0_S2SBR field descriptions

Field	Description
0 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
1–15 S2SBT	Slot 2 Start Boundary Time This field specifies the time in intervals of 1 μs, occurring after the rising edge of the timing sync pulse, at which the slot 2 should start.

47.4.30 Slot 3 Start Boundary Register (PSI5_CH0_S3SBR)

The following figure shows the Slot 3 Start Boundary Register.

Address: 0h base + 194h offset = 194h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Read	0	S3SBT														
Write																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

PSI5_CH0_S3SBR field descriptions

Field	Description
0 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
1–15 S3SBT	Slot 3 Start Boundary Time This field specifies the time in intervals of 1 μ s, occurring after the rising edge of the timing sync pulse, at which the slot 3 should start.

47.4.31 Slot 4 Start Boundary Register (PSI5_CH0_S4SBR)

The following figure shows the Slot 4 Start Boundary Register.

Address: 0h base + 196h offset = 196h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Read	0	S4SBT														
Write																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

PSI5_CH0_S4SBR field descriptions

Field	Description
0 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
1–15 S4SBT	Slot 4 Start Boundary Time This field specifies the time in intervals of 1 μ s, occurring after the rising edge of the timing sync pulse, at which the slot 4 should start.

47.4.32 Slot 5 Start Boundary Register (PSI5_CH0_S5SBR)

The following figure shows the Slot 5 Start Boundary Register.

Address: 0h base + 198h offset = 198h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Read	0	S5SBT														
Write																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

PSI5_CH0_S5SBR field descriptions

Field	Description
0 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
1–15 S5SBT	Slot 5 Start Boundary Time This field specifies the time in intervals of 1 μs, occurring after the rising edge of the timing sync pulse, at which the slot 5 should start.

47.4.33 Slot 6 Start Boundary Register (PSI5_CH0_S6SBR)

The following figure shows the Slot 6 Start Boundary Register.

Address: 0h base + 19Ah offset = 19Ah

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Read	0	S6SBT														
Write																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

PSI5_CH0_S6SBR field descriptions

Field	Description
0 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
1–15 S6SBT	Slot 6 Start Boundary Time This field specifies the time in intervals of 1 μs, occurring after the rising edge of the timing sync pulse, at which the slot 6 should start.

47.4.34 Slot *n* End Boundary Register (PSI5_CH0_SnEBR)

The following figure shows the Slot *n* Start Boundary Register (PSI5_S *n* SBR).

Slot *n* End Boundary register is a single register which can be programmed to define the "end" of the *n*th slot. The *n* can be defined in the SLOT_NO field of this register.

There are seven boundary registers for the six slots. Six of these registers (S1SBR, S2SBR, S3SBR, S4SBR, S5SBR, and S6SBR) are used to define the start of each of the six slots. The seventh register SnEBR (Slot *n* End Boundary register) defines the end of the *n*th slot. The *n* is programmed in the slot_no field of this register.

The number of slots would automatically be available from the SnEBR. The SnEBR contains the slot number of the last slot and the end boundary of the same. Each slot can separately be enabled or disabled by using the slot_en bit in the SnFCR. When the start slot boundaries are properly defined in the SnSBR (Slot *n* Start Boundary Register) register but the slot_en bit = 0 in the corresponding SnFCR (Slot *n* Frame Configuration register) register, then the corresponding slot is treated as "available" but "unconfigured." For such a slot the data/diagnostic bits all remain 0. The "T" bit will, however, get set if a message that was supposed to come in a configured slot actually arrives in an unconfigured slot. Note that the various parameters like the Data length/CRC configuration and so on are valid ONLY for the configured slots. Even when sampling a message in an unconfigured slot, these parameters would be taken from the previous/next configured slot only. It is assumed that no data would be sent in an unconfigured slot. If a data is received in this unconfigured slot it would never be treated as belonging to the unconfigured slot. As a special case if "S0" and "S1" are detected in different slots then a T-bit error is set .

NOTE

The start slot boundaries in the S *n* SBR should be correctly programmed. If this is not done, the correct operation cannot be guaranteed. Slots occurring after the slot_no value programmed in the SnEBR are treated as being unavailable (that is, the data in all the slots occurring after the SnEBR is rejected by the Manchester decoder).

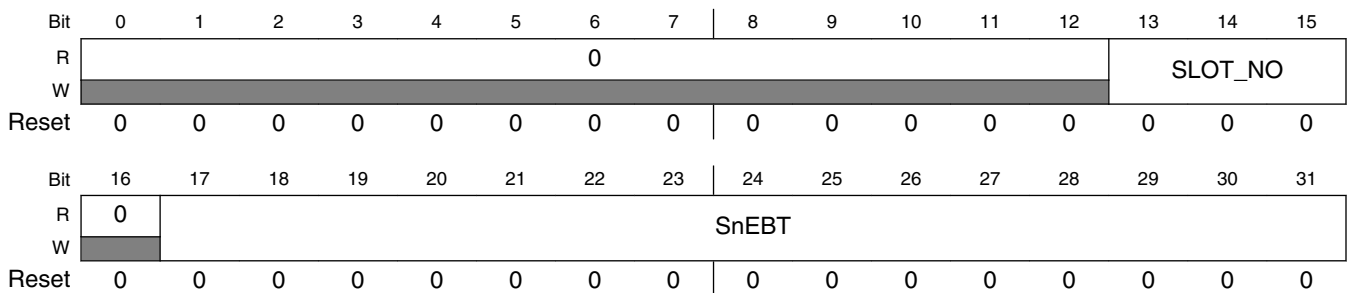
Following are the examples:

- If only three slots are required between two sync pulses, program S1SBR, S2SBR, and S3SBR with the correct value of the start boundaries of the three slots and

program slot_no = 3 in the S *n* EBR and define the end boundary of the third slot in S *n* EBR. The design ignores slots 4, 5, and 6.

- For six slots, the required programming would be: program the start of each of the six slots in S1SBR, S2SBR, S3SBR, S4SBR, S5SBR, and S6SBR. Then program the end of the sixth slot in the S *n* EBR by putting slot_no = 6 and define the end boundary of the sixth slot in S *n* EBR.

Address: 0h base + 19Ch offset = 19Ch



PSI5_CH0_SnEBR field descriptions

Field	Description
0–12 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
13–15 SLOT_NO	The field indicates the Slot number{1-6} for which Slot End Boundary Time is defined. Any value other than 1-6 is defaulted to 0. A value 0 here would mean that the manchester decoder would not sample any slots. This slot number also defines the number of slots that the IP assumes to exist between 2 sync pulses.
16 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
17–31 SnEBT	Slot <i>n</i> End Boundary Time. This field specifies the time in intervals of 1μs, occurring after the rising edge of timing sync pulse, at which the slot 'n' should end.

47.4.35 Data Output Block Configuration Register (PSI5_CH0_DOBCR)

The following figure shows the Data Output Block Configuration Register.

Table 47-118. CMD_TYPE details

CMD_TYPE	PSI5 Frame ¹	Effect on registers	Writeable Register lengths
"000" - 0	Short Frame(V1.3) with 31 "1s" as the start condition	Command can be written to DPR for auto calculation of stuff/start/CRC bits. Can always be overwritten in DSR/DBR registers provided DSR_RDY == 1 or DBR_RDY == 1 respectively	6 bits in DPR 15bits ² in DBR/DSR.
"001" - 1	Short Frame(V1.3) with 5 "0s" as the start condition	-do-	-do-
"010" - 2	Long Frame(V1.3) with 31 "1s" as the start condition	-do-	16bits in DPR. 29bits ² in DBR/DSR.
"011" - 3	Long Frame(V1.3) with 5 "0s" as the start condition	-do-	-do-
"100" - 4	X-Long Frame(V1.3) with 31 "1s" as the start condition	-do-	22bits in DPR. 37bits ² in DBR/DSR
"101" - 5	X-Long Frame(V1.3) with 5 "0s" as the start condition.	-do-	-do-
"110" - 6	XX-Long (V2.0) -	-do-	24 bits in DPR. 43bits ² in DBR/DSR
"111" - 7	Non Standard Length.	Command can only be written to DSR and DBR registers provided DSR_RDY == 1 or DBR_RDY == 1 respectively	Programmable from 1 to 64 bits in DBR/DSR, depending on the value of PSI5_DBOCR[DATA_LENGTH] field.

1. The start condition(31 "1s" or 5 "0s") is automatically taken care by the hardware and should NOT be written to the registers as part of the command.
2. This is also the length of the command that will be shifted out during the ECU to Sensor communication, regardless of how many "actual" bits are written to DBR/DSR.

Table 47-119. DOBCR bit configurations and the related output states

Output path states	GTM_TRIG_SEL	SP_PULSE_SEL	OP_SEL
State1 ¹	0	1	1
State2 ¹	0	0	1
State3 ¹	1	1	1
State4 ¹	1	0	1
State5 ¹	1	x	0

1. These states are shown in [Valid states for integrated sync pulse generator](#). In State5, the pulse width and period is to be controlled directly from the GTM. In this state, the PW0/1D registers have no effect on the Pulse Width Modulation, when observed on the "sdout" pin.

NOTE

Please refer to [Figure 47-147](#) for the details about the above bits. [Figure 47-147](#) is the actual implementation of [Figure](#)

47-146 such that various contentions between the switches (S1, S2, and so on) do not happen.

Address: 0h base + 1A0h offset = 1A0h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Read	DATA_LENGTH[5:0]					DBR_RST	DSR_RST		CMD_TYPE			DEFAULT_SYNC	GTM_TRIG_SEL	SP_PULSE_SEL	OP_SEL	SW_READY
Write																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

PSI5_CH0_DOBCR field descriptions

Field	Description
0–5 DATA_LENGTH[5:0]	Can take on values from 0 to 63 corresponding to 1bit to 64bit non standard length commands. When PSI5_DOBCR[CMD_TYPE] == 7(Non Standard) , then this field controls the length of command that can be written to the DBR and the DSR registers, else the value in this field is ignored. This when DATA_LENGTH = 0, then the length of the command that can be written to DSR and the DBR is 1, when DATA_LENGTH = 1, then the length of the command that can be written to DSR and the DBR is 2 and so on, until a maximum of 64 bit length command when DATA_LENGTH = 63. Note that the length of the command that is programmed by using this field, is also equal to the length of the command that shifts out of the DSR register. In other words, it is equal to the number of bits that shift out of DSR register once the DSR_RDY goes low . After these many shifts have happened , then the value corresponding to "DEFAULT_SYNC" starts getting shifted out and DSR_RDY goes as "1".
6 DBR_RST	This is to reset and reject current content to Data Buffer Register. When this bit is written as "1" then the contents of the DBR are reset to all "0s" (if DEFAULT_SYNC == 0) or all "1s" (if DEFAULT_SYNC == 1). As soon as content is reset the DBR would be ready for new data. Reading this bit would always return a "0". Writable to 1 in one shot. This bit is writable only in the CONFIG and the Normal modes.
7 DSR_RST	This is to reset and reject current content to Data Shift Register. When this bit is written as "1" then the contents of the DSR are reset to all "0s" (if DEFAULT_SYNC == 0) or all "1s" (if DEFAULT_SYNC == 1). As soon as content is reset the DSR would be ready for new data. Reading this bit would always return a "0". Writable to 1 in one shot. This bit is writable only in the CONFIG and the Normal modes.
8–10 CMD_TYPE	These 3 bits indicate the type of command that needs to be transmitted during the ECU to sensor communication. Table 47-118 is a brief description of the same.
11 DEFAULT_SYNC	When this bit is set to "0" then the default value of DSR and DBR registers are all "0s"; when this bit is set to 1 then the default value of DBR and DSR registers are all "1s". This value is used as the value of the default sync pulses shifted at the output path when the DSR_RDY == 1.
12 GTM_TRIG_SEL	GTM event triggered/internal sync pulse generator selection as shift clock for the DSR. 0 Internal sync pulse generator shift triggered 1 GTM event shift triggered
13 SP_PULSE_SEL	Selects the source for the short pulse PWM module: 0 SP module gets the data from the DSR 1 SP module directly gets input from the GTM_event/internal sync pulse generator

Table continues on the next page...

PSI5_CH0_DOBCR field descriptions (continued)

Field	Description
14 OP_SEL	This bit selects as to which would be driving source of the "ipp_do_psi5_sdout" port. 0 The sync pulse generator select as per the Bit3 1 PWM output
15 SW_READY	When this bit is written to 1 the transfer from DBR to DSR automatically happens as soon as DSR_RDY becomes 1. When this bit is kept to 0 then the transfer from DBR to DSR will remain pending . Once the software makes this bit as "1" and DSR_RDY also goes as "1" the transfer to DSR takes place. However, note that when this bit is 0, and the DSR_RDY becomes 1, still the transfer to the DSR will NOT happen. Only when this bit is a 1 , will the transfer happen. This bit can be written in CONFIG and the Normal modes.

1. The start condition(31 "1s" or 5 "0s") is automatically taken care by the hardware and should NOT be written to the registers as part of the command.
2. This is also the length of the command that will be shifted out during the ECU to Sensor communication, regardless of how many "actual" bits are written to DBR/DSR.
3. These states are shown in [Valid states for integrated sync pulse generator](#). In State5, the pulse width and period is to be controlled directly from the GTM. In this state, the PW0/1D registers have no effect on the Pulse Width Modulation, when observed on the "sdout" pin.

47.4.36 Manchester Decoder Disable Offset (PSI5_CH0_MDDIS_OFF)

The Manchester Decoder Disable Offset register (PSI5_MDDIS_OFF) defines the time in intervals of 1 μ s, for which the Manchester decoder remains disabled after the falling edge of the sync pulse. Using this register, this time can be programmed from 0 μ s to 128 μ s. This works on the 1 μ s clock from the common clock generator module.

Address: 0h base + 1A2h offset = 1A2h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Read	0								MDDIS_OFF							
Write																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

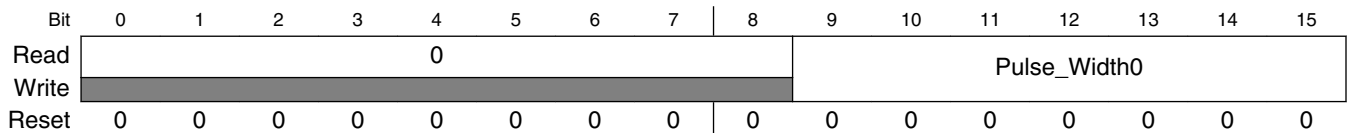
PSI5_CH0_MDDIS_OFF field descriptions

Field	Description
0–8 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
9–15 MDDIS_OFF	These 7 bits can be used to program the time for which the Manchester Decoder remains disabled. This offset is added AFTER the falling edge of the sync pulse. Thus the total time for which the manchester decoder remains disabled = TsyncH + MDDIS_OFF where TsyncH is the high time for the sync pulse . Thus TsyncH = Pulse_Width0 or Pulse_Width1 as the case maybe, depending on whether a "0" is being pulse modulated or a "1". Note that the Manchester Decoder will ALWAYS remain disabled during the high time of the Sync Pulse. Using this register it is possible to disable the Manchester Decoder BEYOND the Sync Pulse High Time as well. Figure 47-151 shows the "MDDIS_OFF" definition.

47.4.37 Pulse Width for Data Bit Value 0 (PSI5_CH0_PW0D)

The following figure shows the Pulse Width for Data Bit 0 Register.

Address: 0h base + 1A4h offset = 1A4h



PSI5_CH0_PW0D field descriptions

Field	Description
0–8 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
9–15 Pulse_Width0	This defines the width (in μ s) of data value '0' to be send from Data Output Register. It is the number of clock cycles(1 μ s clock) counted upto width "Pulse_Width0", as soon as trigger appears from the ISPG(Internal Sync Pulse Generator) or the GTM . Can take max value of 127. When using PSI5 version 1.3 ECU-to-sensor Communication Pulse_Width0 should be configured to 0. Figure 47-151 shows the definition of "Pulse_Width0".

47.4.38 Pulse Width for Data Bit Value 1 (PSI5_CH0_PW1D)

The following figure shows the Pulse Width for Data Bit 1 Register.

NOTE

When using the Version 1.3 format for ECU-to-sensor communication, the pulse width is NOT disabled. In this case, PW0D has to be programmed as 0. PW1D has to have the value of the length of the pulse desired when a 1 has to be transferred.

Address: 0h base + 1A6h offset = 1A6h



PSI5_CH0_PW1D field descriptions

Field	Description
0–8 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

Table continues on the next page...

PSI5_CH0_PW1D field descriptions (continued)

Field	Description
9–15 Pulse_Width1	This defines the width (in μ s) of data value '1' to be send from Data Output Register. It is the number of clock cycles(1 μ s clock) counted upto width "Pulse_Width1", as soon as trigger appears from the ISPG(Internal Sync Pulse Generator) or the GTM . Can take max value of 127. Figure 47-151 shows the definition of "Pulse_Width1".

47.4.39 Counter Target Pulse Register (PSI5_CH0_CTPR)

The following figure shows the Counter Target Pulse Register.

NOTE

Please see [Internal SYNC Pulse generation and coordination across PSI5 channels](#) for details about the Sync Pulse generation and coordination.

Address: 0h base + 1A8h offset = 1A8h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Read	CTPR															
Write																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

PSI5_CH0_CTPR field descriptions

Field	Description
0–15 CTPR	Counter Target Pulse Register. This is the target counter value. Once the CTC (channel target counter) reaches the value of CTPR this CTC is reset. CTC is a 16 bit free running counter that starts/stops depending on the CTC_GED (in GCR) bit or the CTC_ED bit (in PCCR). It gets reset initially when its value reaches CIPR and subsequently when its value reaches CTPR. The fact that which out of CTC_GED or the CTC_ED bit enables/disables it, is dependent on the CTC_GED_SEL bit in the PCCR. When CTC_GED/CTC_ED == 0 then the CTC is reset. The minimum value of CTPR is 6. Values less than 6 will be defaulted to 0 and the CTPR counter will not start. The value in CTPR indicates the Time Period of the internally generated sync pulses. The un modulated (No PWM) internally generated sync pulses have a high time of 4-sp_ts_clk cycles i.e. the duty cycle of the un modulated internally generated sync pulses is (4/CTPRI *100). The 4 sp_ts_clk cycles is just the nascent value of the Pulse which actually triggers the PWD counters Note that this gives one more possible state, in addition to the states shown in Valid states for integrated sync pulse generator . This additional state allows the output of the internal sync pulse generator to be made directly available at the output, without being pulse modulated. The CTPR describes ONLY the SYNC pulse period and not the sync pulse length which is governed by the PWD registers, except in State5 where in the GTM governs these parameters.

47.4.40 Counter Initialize Pulse Register (PSI5_CH0_CIPR)

The following figure shows the Counter Initialize Pulse Register.

NOTE

Due to internal Flop synchronisation there can be a jitter of +/- 1 ns between the offset times that are programmed in the CIPR of different channels and the ones actually being observed. Similarly there can be a jitter of +/- 1 ns in the programmed CTPR and the measured CTPR.

NOTE

Please see [Internal SYNC Pulse generation and coordination across PSI5 channels](#) for details about the Sync Pulse generation and coordination.

Address: 0h base + 1AAh offset = 1AAh

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Read	CIPR															
Write																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

PSI5_CH0_CIPR field descriptions

Field	Description
0–15 CIPR	Counter initialize pulse register. This is the value using which the CTC gets reset, the first time after which it is enabled using CTC_GED (in GCR) or the CTC_ED bit (in PCCR). All subsequent resets are done when the CTC value reaches CTPR. The CIPR is used to set the offset time between the start of sync pulses between two different PSI5 channels, when the CTC of both of these channels are started simultaneously using CTC_GED. The offset between any two PSI5 channels would be CIPR_channel1 - CIPR_channel2 sp_ts_clk cycles. This if channel 1 has a CIPR = 0 while channel2 has CIPR = 10, then the offset would be 10 sp_ts_clk cycles.

47.4.41 Data Preparation Register Low (PSI5_CH0_DPRL)

The following figure shows the Data Preparation Register Low.

This register is CPU writable only in Normal mode and Disable mode.

This register is the lower register used for writing the standard data and the address commands and is to be used for writing standard length ECU-to-sensor commands. The CRC/stuff/start bits are automatically appended by the hardware before the command is

transferred to DBR. The commands to this register are accepted ONLY when $\text{PSI5\_GISR}[\text{DPR\_RDY}] == 1$. Whenever $\text{PSI5\_GISR}[\text{DPR\_RDY}] == 1$, then each bit of this register defaults to 0. Writing to this register when $\text{PSI5\_GISR}[\text{DPR\_RDY}] == 0$ will cause the $\text{PSI5\_GISR}[\text{IS\_PROW}]$ status bit to be set and the generation of an error interrupt, if it is enabled. Further, trying to write to this register when $\text{PSI5\_DOBCR}[\text{CMD\_TYPE}] = "7"$ results in the setting of the $\text{PSI5\_GISR}[\text{IS\_PROW}]$ bit and the resulting write is rejected.

Table 47-126. DPR details

$\text{PSI5\_DOBCR}[\text{CMD\_TYPE}]$	Standard PSI5 FRAME Types	Writeable bits in PSI5_DPRL
"0" or "1"	Short Frame(PSI5 V1.3)	6($\text{DPR}[5:0]$)
"2" or "3"	Long Frame(PSI5 V1.3)	16($\text{DPR}[15:0]$)
"4" or "5"	X-Long Frame(PSI5 V1.3)	22($\text{DPR}[21:0]$)
"6"	XX-Long Frame(PSI5 V2.0)	24($\text{DPR}[23:0]$)
"7"	Not Available for write	Not Available for write

Address: 0h base + 1ACh offset = 1ACh

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0								DPR																							
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	

PSI5_CH0_DPRL field descriptions

Field	Description
0–7 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
8–31 DPR	$\text{DPR}[23:0]$ are the 24-bits of the DPR register used for writing the variable length, standard ECU-to-Sensor commands, comprising of the address, data and other fields. Note that the IP is transparent to the arrangement of the address, data and other fields and it treats the bits in these fields identically for the purpose of CRC calculation and transmission. This register is writeable only when using the standard length PSI5 frames. For details about CRC calculation please refer to Data transmission . The writeable bits in this register are governed by the $\text{PSI5\_DOBCR}[\text{CMD\_TYPE}]$ bit fields as per Table 47-126 . Note that the unwriteable bits in this register field are always read as "0s". This register has to be written by the CPU.

47.4.42 Data Preparation Register High (PSI5_CH0_DPRH)

Address: 0h base + 1B0h offset = 1B0h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	Reserved																															
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	

PSI5_CH0_DPRH field descriptions

Field	Description
0–31 Reserved	This field is reserved. This register is always read as 0s. Writing to this register does not have any effect.

47.4.43 Data Buffer Register Low (PSI5_CH0_DBRL)

The following figure shows the Data Buffer Register Low.

This register is CPU writable only in Normal mode and Disable mode.

NOTE

When DEFAULT_SYNC == 0, then default value of this register is 00000000. When DEFAULT_SYNC == 1, then the default value of this register is FFFFFFFF. Note that the default value is different from the reset value. The reset value of the register is always "00000000". The default value is loaded once the IP enters the Normal mode.

NOTE

Further, the commands to this register are accepted ONLY when PSI5_GISR[DBR_RDY] == 1. Whenever PSI5_GISR[DBR_RDY] == 1, then each bit of this register defaults to DEFAULT_SYNC value. Writing to this register when PSI5_GISR[DBR_RDY] == 0 will set the status bit PSI5_GISR[IS_BROW] and generate an error interrupt, if it is enabled.

Table 47-129. DBRL[DBR] details

PSI5_DBCR [CMD_TYPE]	DBR bits	Bit Assignment
"7" (Non Standard Length)	DBR[31:0]]	Variable length of writeable bits , as per the length programmed in the PSI5_DBCR[DATA_LENGTH]
"6"(XX Long)	DBR[31:30]	DPR[19:18]
	DBR[29]	1'b0(stuff)
	DBR[28:23]	DPR[17:12]
	DBR[22]	1'b0(stuff)
	DBR[21:16]	DPR[11:6]
	DBR[15]	1'b0(stuff)
	DBR[14:9]	DPR[5:0]
	DBR[8:0]	"011111110"

Table continues on the next page...

Table 47-129. DBRL[DBR] details (continued)

PSI5_DOBCR [CMD_TYPE]	DBR bits	Bit Assignment
"4" or "5" (X-Long)	DBR[31]	1'b1(stuff)
	DBR[30:28]	DPR[20:18]
	DBR[27]	1'b1(stuff)
	DBR[26:24]	DPR[17:15]
	DBR[23]	1'b1(stuff)
	DBR[22:20]	DPR[14:12]
	DBR[19]	1'b1(stuff)
	DBR[18:16]	DPR[11:9]
	DBR[15]	1'b1(stuff)
	DBR[14:12]	DPR[8:6]
	DBR[11]	1'b1(stuff)
	DBR[10:8]	DPR[5:3]
	DBR[7]	1'b1(stuff)
	DBR[6:4]	DPR[2:0]
	DBR[3]	1'b1(stuff)
	DBR[2:0]	"010"(start seq)
"2" or "3" (Long)	DBR[31:29]	"000"
	DBR[28]	CRC[0]
	DBR[27]	1'b1(stuff)
	DBR[26:24]	{CRC[1],CRC[2],DPR[15]}
	DBR[23]	1'b1(stuff)
	DBR[22:20]	DPR[14:12]
	DBR[19]	1'b1(stuff)
	DBR[18:16]	DPR[11:9]
	DBR[15]	1'b1(stuff)
	DBR[14:12]	DPR[8:6]
	DBR[11]	1'b1(stuff)
	DBR[10:8]	DPR[5:3]
	DBR[7]	1'b1(stuff)
	DBR[6:4]	DPR[2:0]
	DBR[3]	1'b1(stuff)
	DBR[2:0]	"010"(start seq)

Table continues on the next page...

Table 47-129. DBRL[DBR] details (continued)

PSI5_DOBCR [CMD_TYPE]	DBR bits	Bit Assignment
"0" or "1" (Short)	DBR[31:15]	0
	DBR[14:12]	{CRC[0],CRC[1],CRC[2]}[
	DBR[11]	1'b1(stuff)
	DBR[10:8]	DPR[5:3]
	DBR[7]	1'b1(stuff)
	DBR[6:4]	DPR[2:0]
	DBR[3]	1'b1(stuff)
	DBR[2:0]	"010"(start seq)

Address: 0h base + 1B4h offset = 1B4h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R																																
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

PSI5_CH0_DBRL field descriptions

Field	Description
0–31 DBR	This register contains the Lower 32 bits(DBR[31:0]) of the max-64 bit length command (DBR[63:0]). The higher bits, DBR[63:32] are contained in the DBRH register. These fields can automatically be updated by the hardware (when the CRC/stuff/start bits are appended to the data in DPRL) or these fields can be written by the CPU(when PSI5_GISR[DBR_RDY] == 1 in the Normal mode).Hardware updation occurs when PSI5_GISR[DPR_RDY]== 0and PSI5_GISR[DBR_RDY] == 1. Depending on the PSI5_DOBCR[CMD_TYPE] value this register can have different data assignments as describe in Table 47-129 . The table (except when PSI5_DOBCR[CMD_TYPE] = "7") show the bit assignment of the DBR[31:0] when these bits are automatically updated by the hardware. When PSI5_DOBCR[CMD_TYPE] == "7"), then only the CPU can write to these registers.

47.4.44 Data Buffer Register High (PSI5_CH0_DBRH)

The following figure shows the Data Buffer Register High.

This register is CPU writable only in Normal mode and Disable mode.

NOTE

When DEFAULT_SYNC == 0 then the default value of this register is 00000000 . When DEFAULT_SYNC == 1 then the default value of this register is FFFFFFFF . Note that the default value is different from the reset value. The reset value of the register is always "00000000". The default value is loaded once the IP enters the Normal mode. Further, the

commands to this register are accepted ONLY when $\text{PSI5\_GISR}[\text{DBR\_RDY}] == 1$. Whenever $\text{PSI5\_GISR}[\text{DBR\_RDY}] == 1$ then each bit of this register defaults to DEFAULT_SYNC value. Writing to this register when $\text{PSI5\_GISR}[\text{DBR\_RDY}] == 0$ will set the status bit $\text{PSI5\_GISR}[\text{IS\_BROW}]$ and generate an error interrupt, if it is enabled.

Table 47-131. DBRH[DBR] details

PSI5_DOBCR [CMD_TYPE]	DBR bits	Bit Assignment
"7" - Non Standard Frame lengths	DBR[63:32]	Variable length of writeable bits , as per the length programmed in the $\text{PSI5\_DOBCR}[\text{DATA\_LENGTH}]$
"6" - XX Long	DBR[63:43]	All 0s
	DBR[42:37]	{CRC[5],CRC[4],CRC[3],CRC[2],CRC[1],CRC[0]}
	DBR[36]	1'b0(stuff)
	DBR[32:35]	DPR[23:20]
"4" or "5" -X Long	DBR[63:37]	All 0s
	DBR[36]	CRC[0]
	DBR[35]	1'b1((stuff)
	DBR[34:32]	{CRC[1],CRC[2],DPR[21]}
"3" or "2"(Long) OR "1" or "0" (Short)	DBR[63:32]	All 0s

Address: 0h base + 1B8h offset = 1B8h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	DBR																															
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	

PSI5_CH0_DBRH field descriptions

Field	Description
0–31 DBR	This register contains the Upper 32 bits(DBR[63:32]) of the max-64 bit length command (DBR[63:0]). The lower bits, DBR[31:0] are contained in the DBRL register. These fields can automatically be updated by the hardware (when the CRC/stuff/start bits are appended to the data in DPRL) or these fields can be written by the CPU(when $\text{PSI5\_GISR}[\text{DBR\_RDY}] == 1$ in the Normal mode).Hardware updation occurs when $\text{PSI5\_GISR}[\text{DPR\_RDY}] == 0$ and $\text{PSI5\_GISR}[\text{DBR\_RDY}] == 1$. Depending on the $\text{PSI5\_DOBCR}[\text{CMD\_TYPE}]$ value this register can have different data assignments as describe in Table 47-131 . Note that when $\text{PSI5\_DOBCR}[\text{CMD\_TYPE}] == "7"$, then only the CPU can write to these registers.

47.4.45 Data Shift Register Low (PSI5_CH0_DSRL)

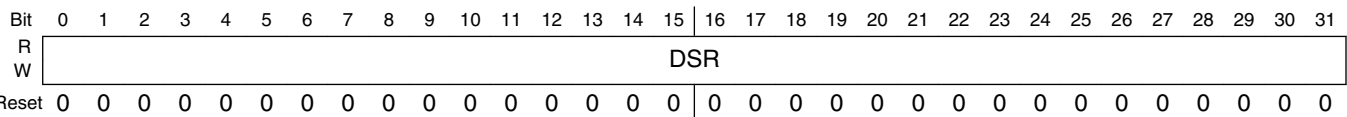
The following figure shows the Data Shift Register Low.

This register is CPU writable only in Normal mode and Disable mode.

NOTE

When DEFAULT_SYNC == 0, then default value of this register is 00000000. When DEFAULT_SYNC == 1, then the default value of this register is FFFFFFFF. Note that the default value is different from the reset value. The reset value of the register is always "00000000". The default value is loaded once the IP enters the Normal mode. Further, the commands to this register are accepted ONLY when PSI5_GISR[DSR_RDY] == 1. Whenever PSI5_GISR[DBR_RDY] == 1 then each bit of this register defaults to DEFAULT_SYNC value. Writing to this register when PSI5_GISR[DSR_RDY] == 0 will set the status bit GISR[IS_DSROW] and generate an error interrupt, if it is enabled. Note that for the hardware updation of DSR with DBR contents, additional condition PSI5_DOBCR[SW_READY] == 1, should also be satisfied in addition to PSI5_GISR[DSR_RDY] == 1 .

Address: 0h base + 1BCh offset = 1BCh



PSI5_CH0_DSRL field descriptions

Field	Description
0–31 DSR	This register contains the lower 32 bits, (DSR[31:0]), of the max-64 bit length command (DSR[63:0]). The higher bits, DSR[63:32], are contained in the DSRH register. These bits can be updated by the hardware or can be written by the CPU. The number of accessible bits in DSR are always equal to the number of accessible bits in DBR, which in turn depend on the value of PSI5_DOBCR[CMD_TYPE] register bits. When these bits are updated by the hardware then there is a one to one correspondence between the bit postions in this register and the bit positions in the DBRL. Thus, when updated by the hardware, then DSR[31:0] = DBR[31:0]. Hardware updation occurs when PSI5_GISR[DSR_RDY]== 1 and PSI5_GISR[DBR_RDY] == 0 and PSI5_DOBCR[SW_READY] == 1. Note that if PSI5_DOBCR[SW_READY] == 0, then the updation of the data from the DBR to DSR will NOT happen even though PSI5_GISR[DSR_RDY] == 1. These bits can be written by the CPU in the Normal mode , when PSI5_GISR[DSR_RDY] == 1'b1.

47.4.46 Data Shift Register High (PSI5_CH0_DSRH)

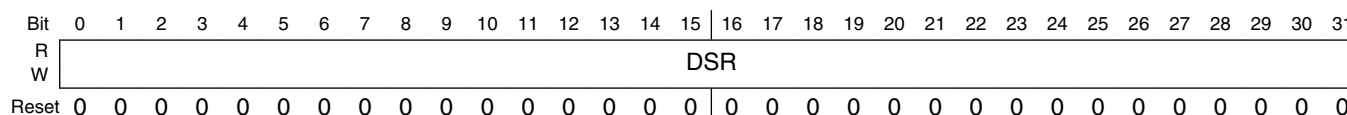
The following figure shows the Data Shift Register High.

This register is CPU writable only in Normal mode and Disable mode.

NOTE

When `DEFAULT_SYNC == 0` then the default value of this register is `00000000`. When `DEFAULT_SYNC == 1` then the default value of this register is `FFFFFFFF`. Further, the commands to this register are accepted ONLY when `PSI5_GISR[DSR_RDY] == 1`. Whenever `PSI5_GISR[DBR_RDY] == 1` then each bit of this register defaults to `DEFAULT_SYNC` value. Writing to this register when `PSI5_GISR[DSR_RDY] == 0` will set the status bit `GISR[IS_DSROW]` and generate an error interrupt, if it is enabled. Note that for the hardware updation of DSR with DBR contents, additional condition `PSI5_DOBCCR[SW_READY] == 1`, should also be satisfied in addition to `PSI5_GISR[DSR_RDY] == 1`.

Address: `0h base + 1C0h offset = 1C0h`



PSI5_CH0_DSRH field descriptions

Field	Description
0–31 DSR	This register contains the higher 32 bits (DSR[63:32]) of the max-64 bit length command (DSR[63:0]). The lower bits, DSR[31:0], are contained in the DSRL register. These bits can be updated by the hardware or can be written by the CPU. The number of accessible bits in DSR are always equal to the number of accessible bits in DBR, which in turn depend on the value of <code>PSI5_DOBCCR[CMD_TYPE]</code> register bits. When these bits are updated by the hardware then there is a one to one correspondence between the bit positions in this register and the bit positions in the DBRH. Thus when updated by the hardware, then <code>DSR[63:32] = DBR[63:32]</code>. Hardware updation occurs when <code>PSI5_GISR[DSR_RDY] == 1</code> and <code>PSI5_GISR[DBR_RDY] == 0</code> and <code>PSI5_DOBCCR[SW_READY] == 1</code>. Note that if <code>PSI5_DOBCCR[SW_READY] == 0</code>, then the updation of the data from the DBR to DSR will NOT happen even though <code>PSI5_GISR[DSR_RDY] == 1</code>. These bits can be written by the CPU in the Normal mode when <code>PSI5_GISR[DSR_RDY] == 1'b1</code>.

47.4.47 Device modes and Register bit accesses

1. The following bits are writable in all modes (Normal, Config, and Disable).

- GCR[GLOBAL_DISABLE_REQ]
 - PCCR[PSI5_CH_CONFIG]
 - PCCR[PSI5_CH_EN]
 - Global Counter Enable/Disable (GCR[CTC_GED]). This is to allow the Sync Pulse Generator to be enabled/disabled independently among those channels that are used for staggering, irrespective of the state of the channels NOT being used for staggering.
2. The following register is writeable only in Normal mode and is initialized to all 0s.
 - DPRL
 3. The following registers are writable ONLY in Normal mode. Initialization in Config mode depends on the value of PSI5_DOBCR[DEFAULT_SYNC] bit. When PSI5_DOBCR[DEFAULT_SYNC] == 1 in Config mode, then the below registers are initialized to all 1s in Normal mode. When PSI5_DOBCR[DEFAULT_SYNC] = 0 in Config mode, then the below registers are initialized to all 0s in Normal mode. Once in Normal mode, these registers can then be programmed for any command that needs to be sent. x
 - DBRH/DBRL
 - DSRH/DSRL
 4. The following registers are writable in Config and Normal modes only:
 - All interrupt control registers
 - All SET registers (the one shot registers which are used to set the error flags for software drivers)
 - DMA Control register
 - Channel Enable/Disable (PCCR[CTC_ED]). Using this it is possible to allow the Sync Pulse Generator to be enabled/disabled in Normal mode. The counter is reset when CTC_ED == 0
 - All w1c bits
 5. Apart from these, all other register bits can be written only in Config mode , including the Data registers(PMRH/PMRL/SFRn), that, in Normal mode, are updated by the hardware only.
 6. Debug mode: If the device is in Normal mode and the debug mode is enabled, then ONLY the registers mentioned in pts. 1, 2, 3, and 4 above can be written by the debugger. Register bits mentioned in pt. 5 above, will not be available for write through the debugger. In case the debugger has to change any register, then the device has to be brought in the Disable -> Config mode. The debugger can, however, read any register in any mode.

47.5 Functional description

The PSI5 is a communication protocol for peripheral inertial, pressure, temperature, and position sensors. These sensors are connected to the ECU by just two wires, using the same lines for power supply and data transmission. The transceiver (present externally) provides a pre-regulated voltage to the sensors and reads in the transmitted sensor data. The following figure shows a point-to-point connection for sensors 1 and 2 and bus configuration for sensors 3 and 4.

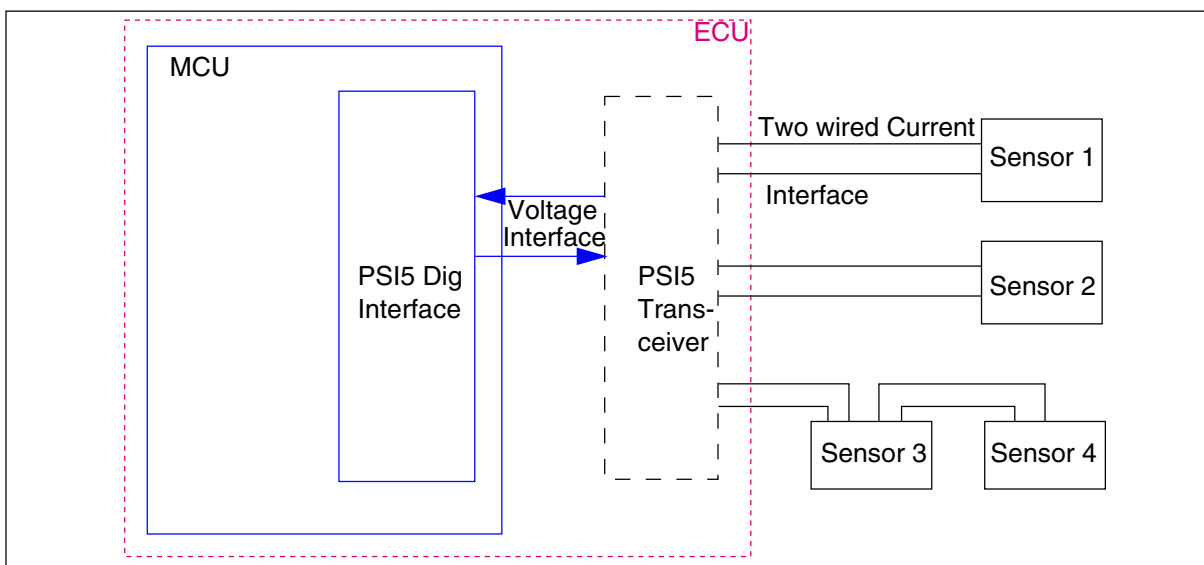


Figure 47-125. System setup (peripheral sensor connected to device)

The ECU (Electronic Control Unit) comprises PSI5 digital interface and the PSI5 transceiver (present externally).

47.5.1 Sensor-to-ECU communication

47.5.1.1 Physical layer

The PSI5 uses two wires to provide both power supply and data transmission to the sensors. The ECU provides a pre-regulated voltage to the sensors. Data transmission from the sensor to the ECU is done by current modulation on the power supply lines. Current oscillations are damped by the ECU.

47.5.1.2 Bit encoding

A "low" level ($I_{S,Low}$) is represented by the Normal (quiescent) current consumption of the sensor(s). A "high" level ($I_{S,High}$) is generated by an increased current sink of the sensor ($I_{S,Low} + \dots I_S$). The current modulation is detected within the PSI5 transceiver.

Manchester coding is used for data transmission. A logic 0 is represented by a rising slope and a logic 1 by a falling slope of the current in the middle of T bit.

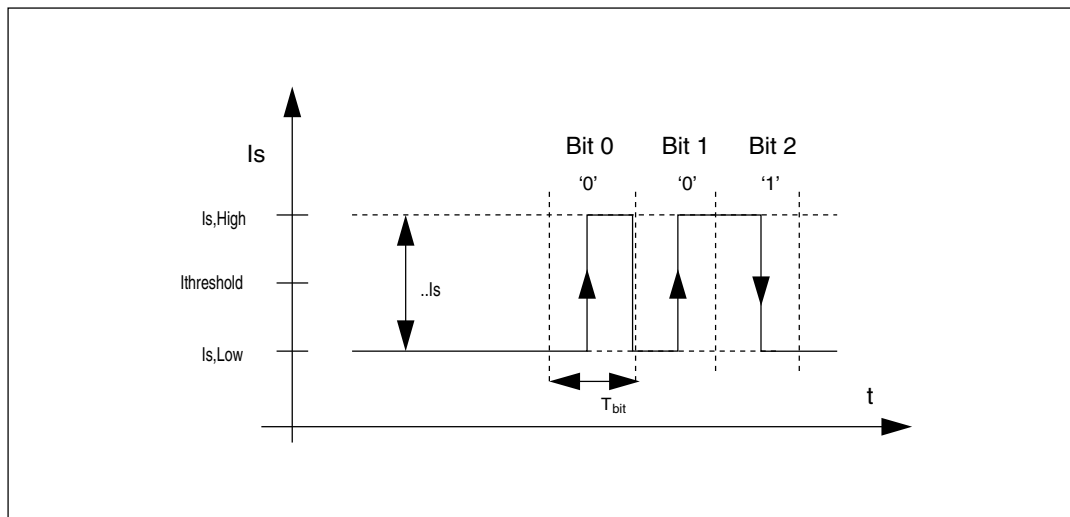


Figure 47-126. Bit encoding of data from sensor

47.5.1.3 Synchronization

For synchronized operation and bus mode, a sync pulse is generated by the PSI5 receiver. The sync pulse is output from the integrated PSI5 receiver to an external PSI5 transceiver that processes these pulses to a pre-regulated voltage level on supply lines to sensors. The voltage change is detected within the sensors.

47.5.2 Data Link Layer

The PSI5 message is received as serial Manchester-coded bits. The bit stream is passed through a Manchester decoder stage that converts it to normal NRZ format bit by bit and stores it serially in the receive register (64 bits). This block clearly identifies start (00) and end of receive frame using preprogrammed frame configuration registers.

- The Manchester-coded bits can be received at the rate of 125 Kbit/s.

- The bit decoding is done using an oversampling and counting algorithm with majority voting and edge recognition. Clocks of $(125 \text{ kHz}) \times 32$ (4 MHz also referred to as $32\times$ clock) are supplied from a common clock generator module. This clock is used for oversampling of the receive bit and thereby decoding it as normal 1 or 0 (NRZ).
- The Manchester decoder is masked (switched off) for the duration when the sync pulse is transmitted. In addition to this, it can be configured to be off for an additional time, after the falling edge of the sync pulse. The duration of this time is configurable in the MDDIS_OFF register.
- It uses a 2003 voter to recognize the encoded half bits.
- There is a 3005 majority voting glitch filter at the data input of the Manchester decoder.

47.5.2.1 Reception of data frames

The extended PSI5 (Power Train Applications) receive data frame consists of p bits containing:

- Two start bits (S1 and S2) {S1=0, S2=0}
- One parity bit (P) with even parity (or 3 CRC bits C0, C1, and C2).
- A data region (D0.....D[$k - 1$]) with $k = 8$ to 28 bits with 1-bit granularity.
- The total length of a PSI5 frame is $p = k + 3$ data bits (in case of frames with parity bit) or $p = k + 5$ data bits (in case of frames with CRC).
- Data bits are received LSB first. The parity or CRC check bits cover the bits of the entire data region.

The length of the data region can vary between $k = 8$ to 28 bits (with 1-bit granularity). The data region can be split into the following fields and regions:

- The bits D0 and D1 can optionally represent Serial Messaging Channel M0, M1.
 - This Serial Messaging Channel (SMC) is used to receive initialization data during fast startup phase or slow acknowledgement/response messages (to command transmitted from ECU requesting data or diagnostic information from the sensor).

Note, that the maximum length of the entire data region is 28 bits. Thus the signal payload region and the optional data fields cannot concurrently be used at their maximum length in some cases.

As the PSI5 module is targeted for powertrain applications, the data frame format differs from standard PSI5 frames for safety applications specified in version 1.3 of the PSI5 specifications. More specifically for supporting the version 2.0 powertrain substandard. The Powertrain data frame format and optional Serial Messaging Channel (SMC) frame format are shown in the following figure.

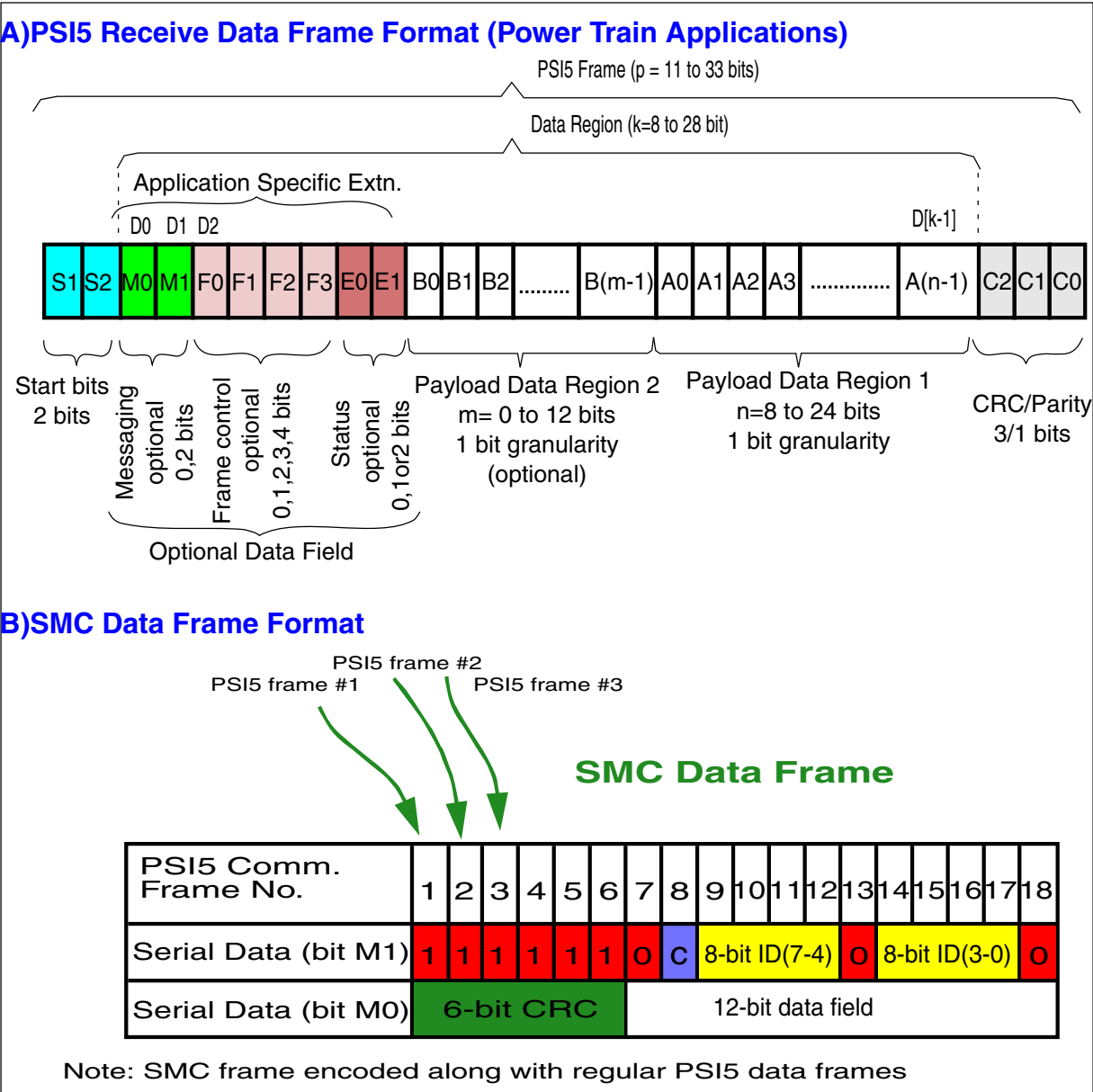


Figure 47-127. Receive data frame format

47.5.2.2 Data Frame Configuration Control

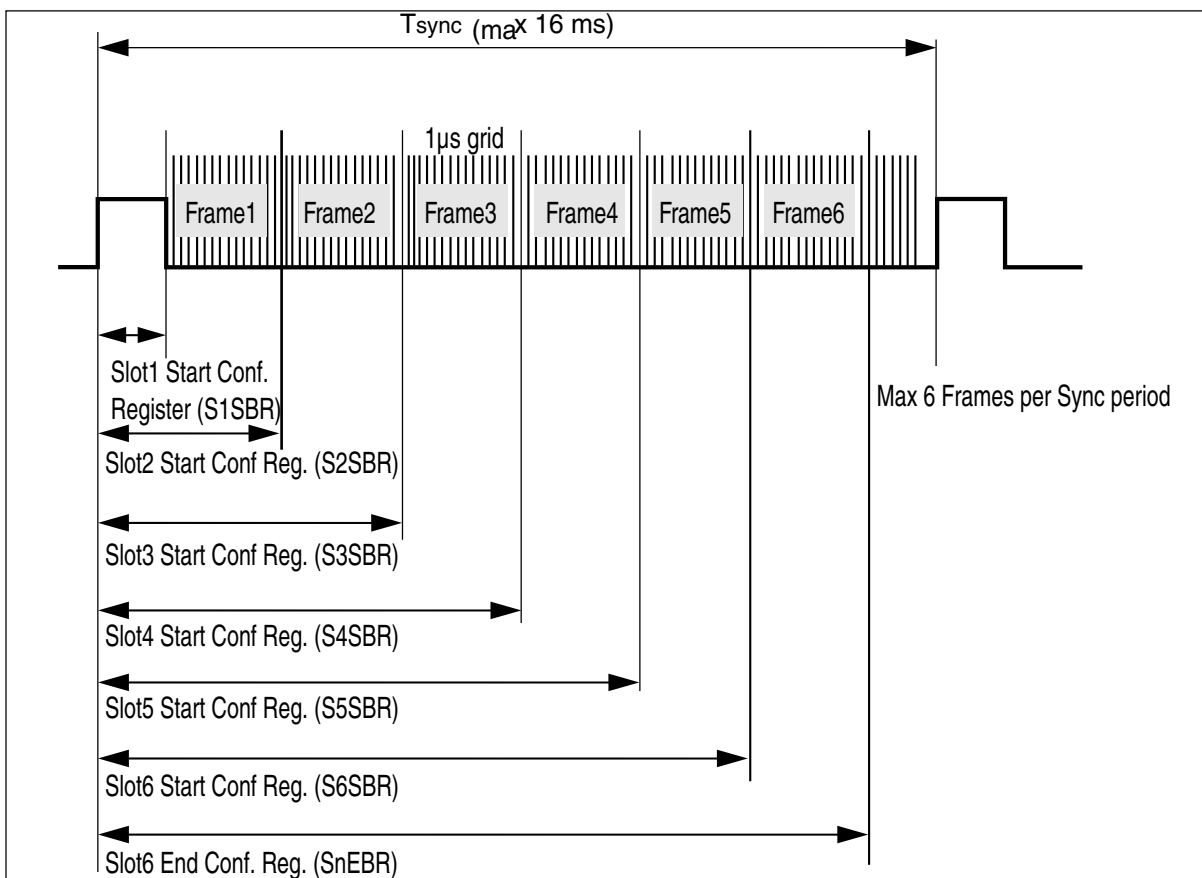


Figure 47-128. Data frame configuration and sampling

At the receive path, as many as six frame slots (reception windows) can be programmed with a start time of consecutive slots (slot1 to slot6) and end time or last slot which can be configured through the Slot *n* Start/End Configuration Registers ([Slot 1 Start Boundary Register \(PSI5_CH0_S1SBR\)](#)-[Slot 6 Start Boundary Register \(PSI5_CH0_S6SBR\)](#) and [Slot n End Boundary Register \(PSI5_CH0_SnEBR\)](#)) there are seven registers for six slots. The slot time value is with respect to timing sync pulse.

The following range is always maintained for T_{sync} (during programming of the GCR and CTPR registers) when timing (sync) pulses are generated through the Sync Pulse Generator.

$$T_{sync} < 16\text{ms}$$

For the maximum defined six frame slots, six frame configuration registers (S1FCR, S2FCR, S3FCR, S4FCR, S5FCR, and S6FCR) are also available. In these registers it is possible to configure the payload region of the expected frames, in addition to other channel-specific parameters.

47.5.2.3 Data frame handling

The following figure shows the detection of the start bits and the various parameters for the same.

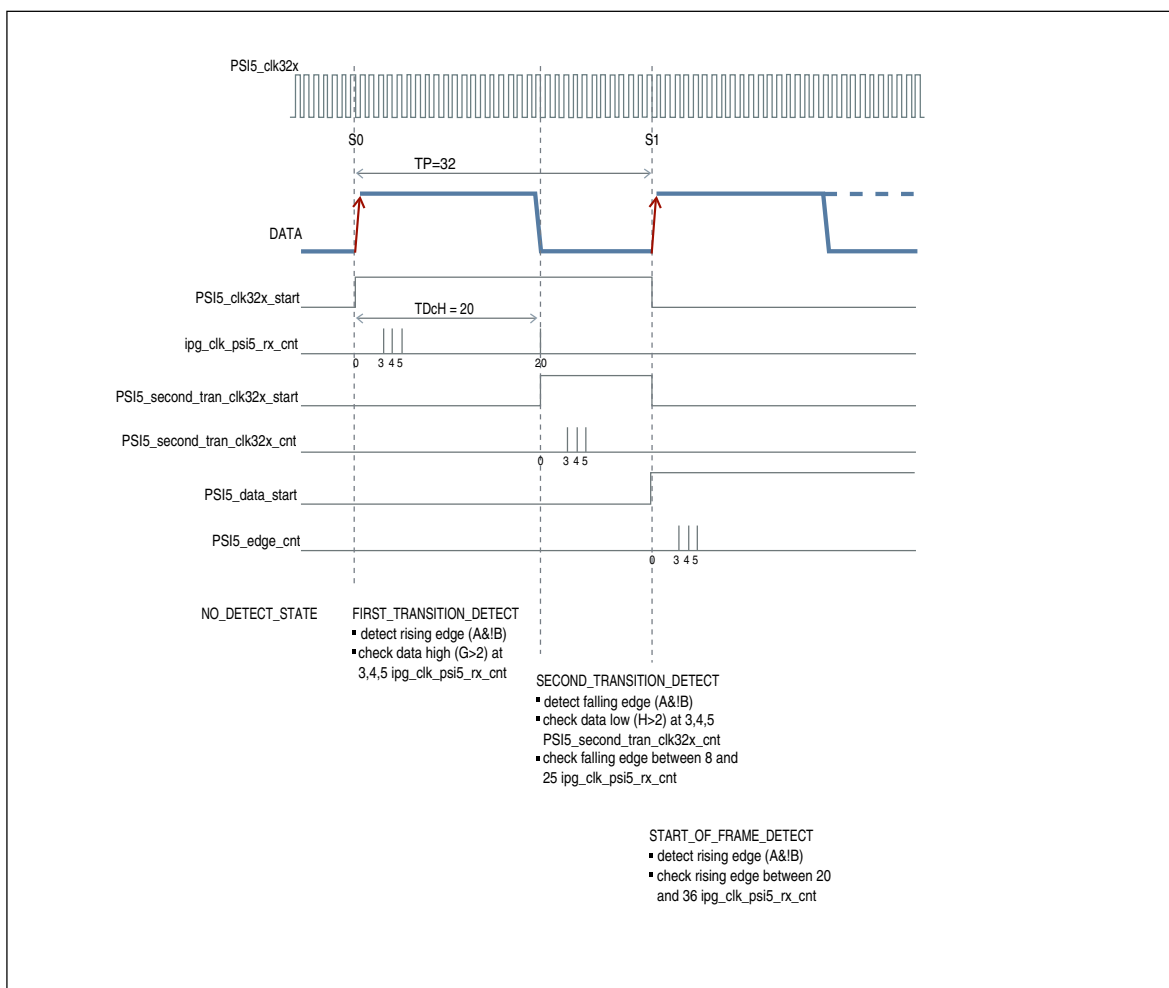
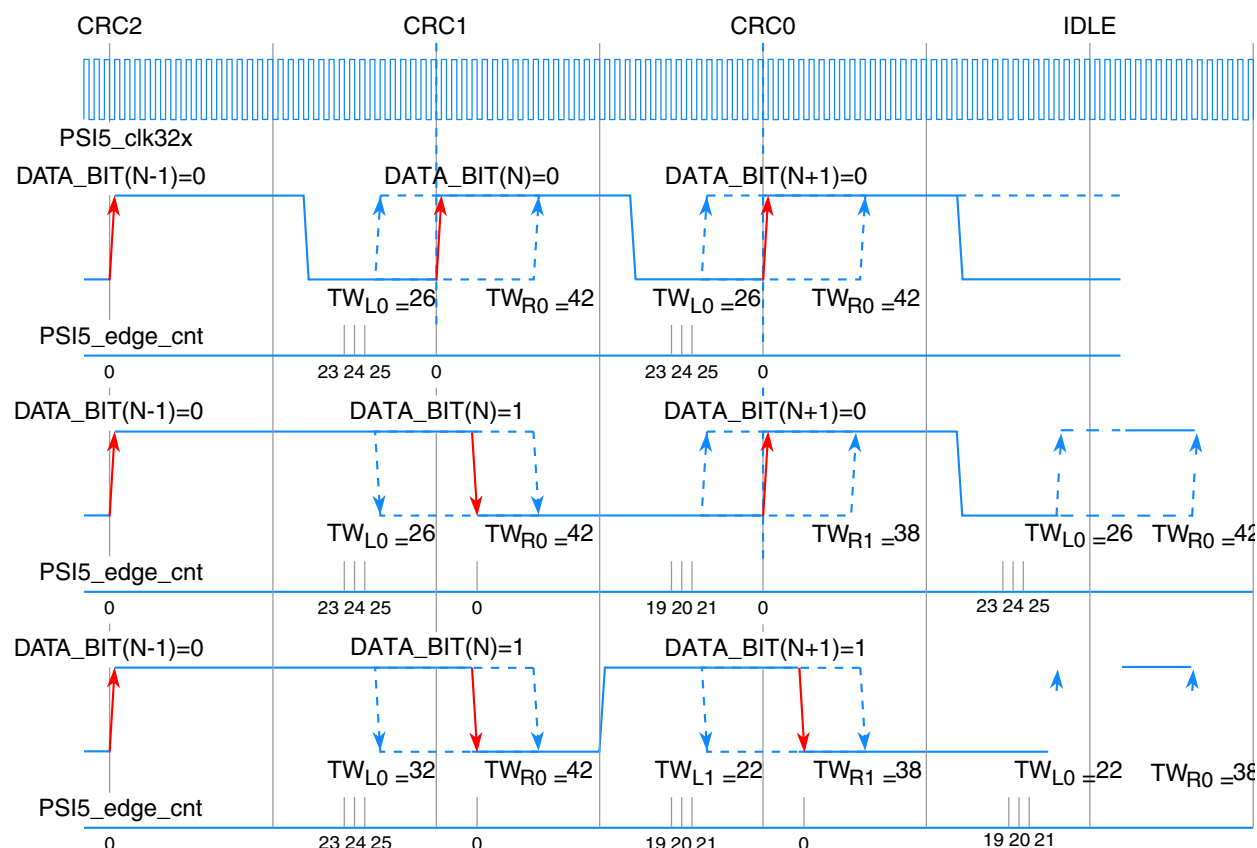


Figure 47-129. Start Bit Detection

The following figure shows how the bits are differentiated into a “1” or a “0” based on the various margins and sample points.



Start Sequence Measured: $T_p = 32$ and $TDcH = 20$

Calculations:

$TDcL = T_p - TDcH = 12$, $\Delta Dc = TDcH - T_p / 2 = 4$; $TMarginL = TDcL / 2 = 6$; $TMarginH = TDcH / 2 = 10$

If $DATA\_BIT(N) = 0$, $TWI = TDcH + TMarginL = 26$; signal sampled at $TWI-3$, -2 and -1

- If signal low, expect rising edge between $TWL0$ and $TWR0$ then $DATA\_BIT(N+1) = 0$.
 $TWL0 = TDcH + TMarginL$ and $TWR0 = T_p + TMarginH$
- If signal high, expect falling edge between $TWL0$ and $TWR0$ then $DATA\_BIT(N+1) = 1$.
 $TWL0 = TDcH + TMarginL$ and $TWR0 = T_p + \Delta Dc + TMarginL$

If $DATA\_BIT(N) = 1$, $TWI = TDcL + TMarginH = 22$; signal sampled at $TWI-3$, -2 and -1

- If signal low, expect rising edge between $TWL1$ and $TWR1$ then $DATA\_BIT(N+1) = 0$.
 $TWL1 = TDcL + TMarginH$ and $TWR1 = T_p - \Delta Dc + TMarginH$
- If signal high, expect falling edge between $TWL1$ and $TWR1$ then $DATA\_BIT(N+1) = 1$.
 $TWL1 = TDcL + TMarginH$ and $TWR1 = T_p + TMarginL$

IDLE state can be reached in two ways:

- Idle can be detected after zero (last data bit). In this the edge is searched in the same window as for data zero i.e. $TWL0 = 26$ to $TWR0 = 42$
- Idle is detected after one (last data bit). In this the edge is searched in the same window as for data one i.e. $TWL0 = 22$ to $TWR0 = 38$

Figure 47-130. Bit Extraction

47.5.2.3.1 Start of frame recognition

47.5.2.3.2 Data state

In the data state, the Manchester decoder samples the number of data bits corresponding to the configured length of the data payload region and 3-bit CRC or 1-bit parity.

47.5.2.3.3 Message diagnostic checks

During the data state, electrical correctness of each bit in regard of Manchester encoding is checked. If an incorrect bit is detected, a 0 is written into the data payload region. Following completion of reception, correctness of the entire message is checked with the CRC and timing of the message reception.

- CRC is recalculated for the data and compared with received CRC bits. The CRC comparison result is stored as a status bit, denoted as the C field in the receive register and the RAM register. Irrespective of the value of the “C” bit the, the frame is always stored in the RAM register and is never discarded.
- If at least one bit is electrically faulty (apart from M0M1), then the bit field E is set to 1 and stored as status bit in the receive register and the RAM register.
- The EM bit is set to 1 if any of M0 or M1 bits are electrically faulty. Note that the E and the EM bits can be set only when a valid start sequence has been detected.
- Set status bit field T in the receive register to 1 if the frame received is in the wrong slot (boundary violations). This bit is NOT set for asynchronous modes.
- Set status bit F if no frame is received in defined frame slot. This bit is NOT set for asynchronous modes.
- The incoming frames after the sync pulse are to be numbered from 1 to 6 (max). This is done by a 3-bit slot counter, which is incremented automatically after the end of each slot irrespective of whether a valid message is received or not. This 3-bit value is stored in the receive register and the RAM registers as the "slot_counter" field.

47.5.2.3.4 T-bit handling

The following paragraphs describe how the IP handles error cases that generate T bit errors. A number of examples are shown, including the error free case, which is used to explain the reception process.

47.5.2.3.4.1 Error free case

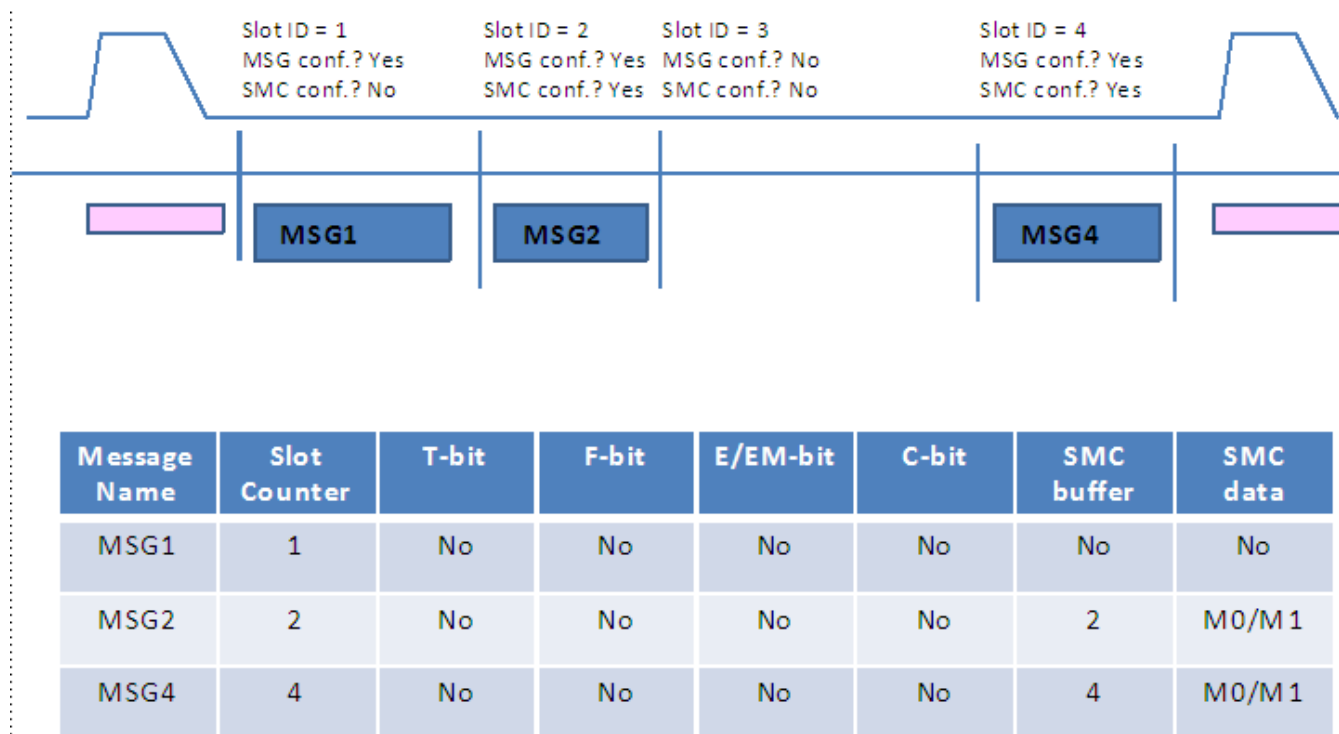


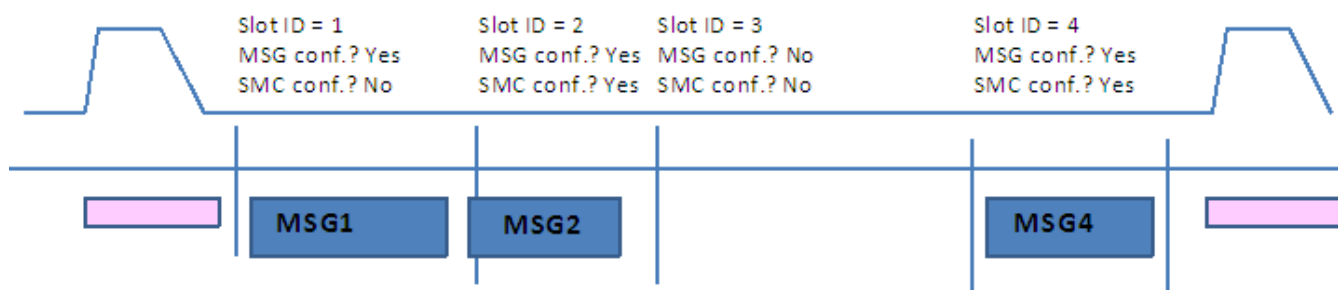
Figure 47-131. Error free case

In this case, all messages are received in the correct slots.

- As soon as the Manchester decoder block is enabled (after the SYNC pulse has finished and any configured disable offset has expired, pink region) the IP checks for the first configured message and starts monitoring the output of the PSI5 transceiver.
- The IP identifies a correct start of message sequence (S0, S1) == '00', and captures the slot counter when the first rising edge of the start sequence is detected. This first rising edge detected is the mid-point transition of the Manchester encoded S0 bit of the message start sequence.
- The expected number of bits configured in the message is received and decoded. Any error in the Manchester encoded bits of the message E bit is set - unless the error is in one of the first two bits and SMC message is configured, in which case the EM bit is set.
- After the final bit is received, a check is made that the message CRC or parity is correct. If the CRC or parity check is not correct, C bit is set for the message.
- Once the CRC checks have been made, the received message — along with the status bits, slot counter in which the start sequence was received, the timestamp of the message, the message data and CRC or parity is saved to the channel message FIFO.

- As soon as the received message has been stored, the idle state of the bus is verified and monitoring of the transceiver output restarts to check for the next configured message.
- If the serial message channel (SMC) is configured, the first two data bits are extracted from a sequence of 7 PSI5 messages as long as no electrical error is detected in those bits (EM error). If the correct start sequence for the SMC message is detected on M1 (6 logical 1s and a logical 0) in those 7 messages, M0 and M1 from the following 11 messages are also extracted and saved in the SMC buffer allocated to the slot in which the PSI5 message is received (in total, 18 PSI5 messages are needed to build one SMC message). When the SMC has been received, the CRC is checked and the status of the check is saved in the SMC buffer. If an EM error occurs after a start sequence has been detected and before the end of the message is reached, M0 and M1 are both saved as 0 and the rest of the message is received, at which point the CRC is checked as normal.

47.5.2.3.4.2 Message Too Early



Message Name	Slot Counter	T-bit	F-bit	E/EM-bit	C-bit	SMC buffer	SMC data
MSG1	1	No	No	No	No	No	No
MSG2	1	Yes	No	No	No	2	M0/M1
MSG4	4	No	No	No	No	4	M0/M1

Figure 47-132. Message Too Early

In this scenario, MSG2 reception starts early, before slot 2 boundary is crossed, and so the slot counter is captured as “1” and because the message starts in the wrong slot, the T bit is set. If no other errors are detected during reception, the assumption is that the

correct MSG2 has been received but not entirely in it's correct slot. The status of the message will include the T bit set and slot counter will be 1, so software can interpret that MSG2 was too early.

The SMC message is received correctly.

47.5.2.3.4.3 Message Too Late

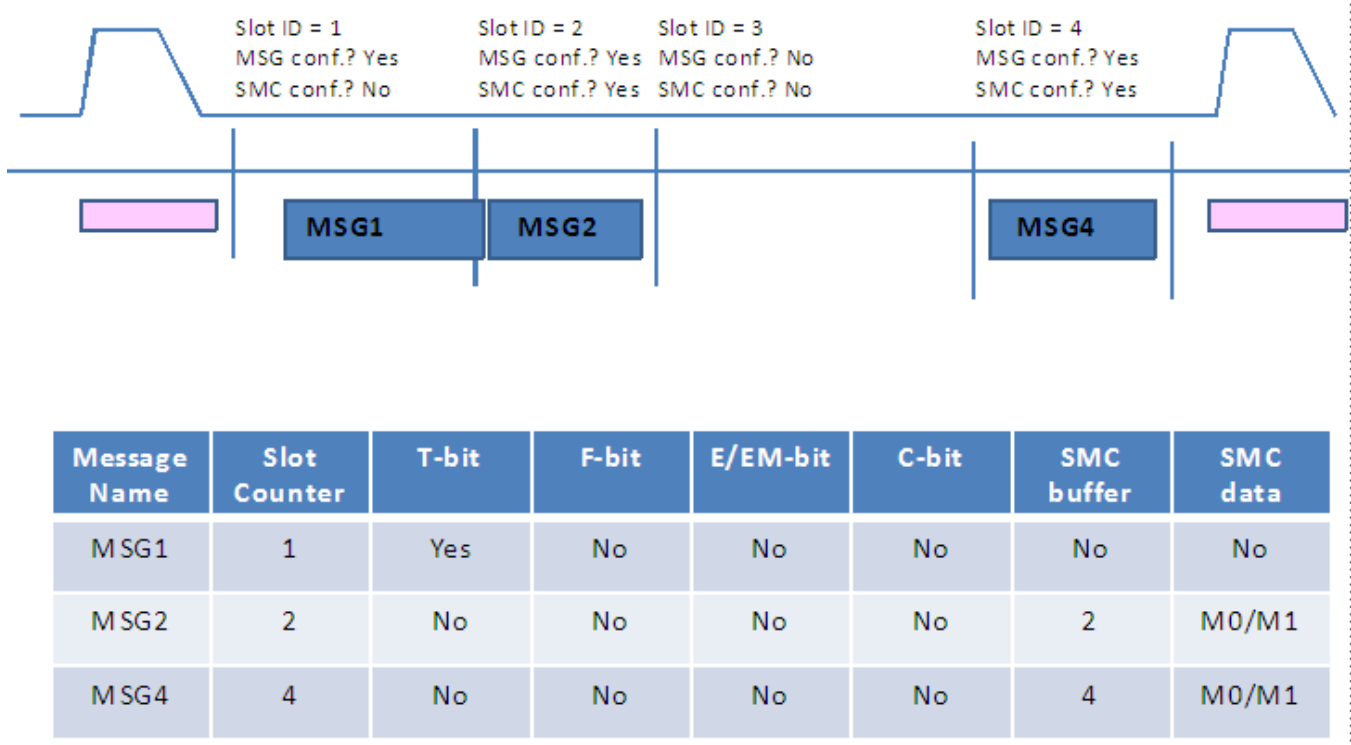
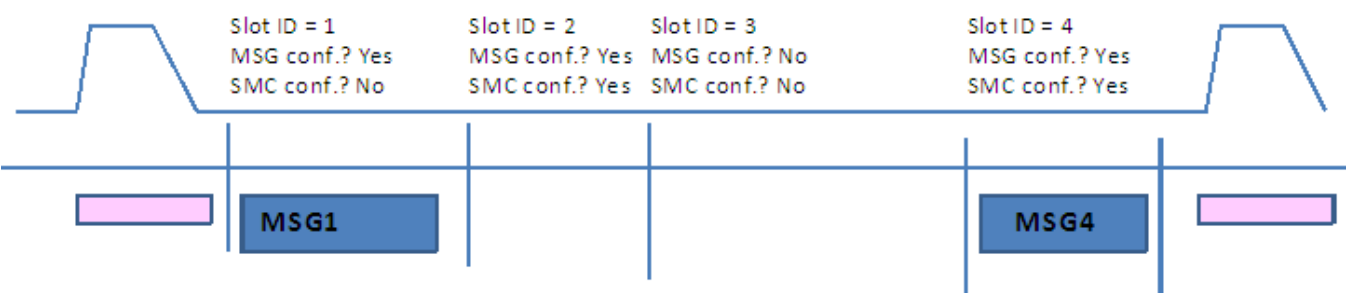


Figure 47-133. Message Too Late

In this scenario, MSG1 reception starts later than expected and ends after slot 2 boundary is crossed. The slot counter is captured as “1” and because the message ends in the wrong slot, the T bit is set. If no other errors are detected during reception, the assumption is that the correct MSG1 has been received but not entirely in it's correct slot. The status of the message will include the T bit set and slot counter will be “1”, so software can interpret that MSG1 was late.

The SMC message is received correctly.

47.5.2.3.4.4 Message Missing



Message Name	Slot Counter	T-bit	F-bit	E/EM-bit	C-bit	SMC buffer	SMC data
MSG1	1	No	No	No	No	No	No
MSG2	2	No	Yes	No	No	2	0/0
MSG4	4	No	No	No	No	4	M0/M1

Figure 47-134. Message Missing

In this scenario, MSG2 is missing.

At the end of slot 2, the IP writes message buffer location 2 (MSG2 location) with slot counter 2 and the F bit set. The data and timestamp regions are filled with 0s.

The IP waits until the start of the next configured slot (slot 4) to update the SMC buffer 2 with “00”. If slot 4 was not configured, the SMC for the missing MSG2 would be updated at the end of the final slot.

47.5.2.3.4.5 Message Too Much Late (in unconfigured slot)

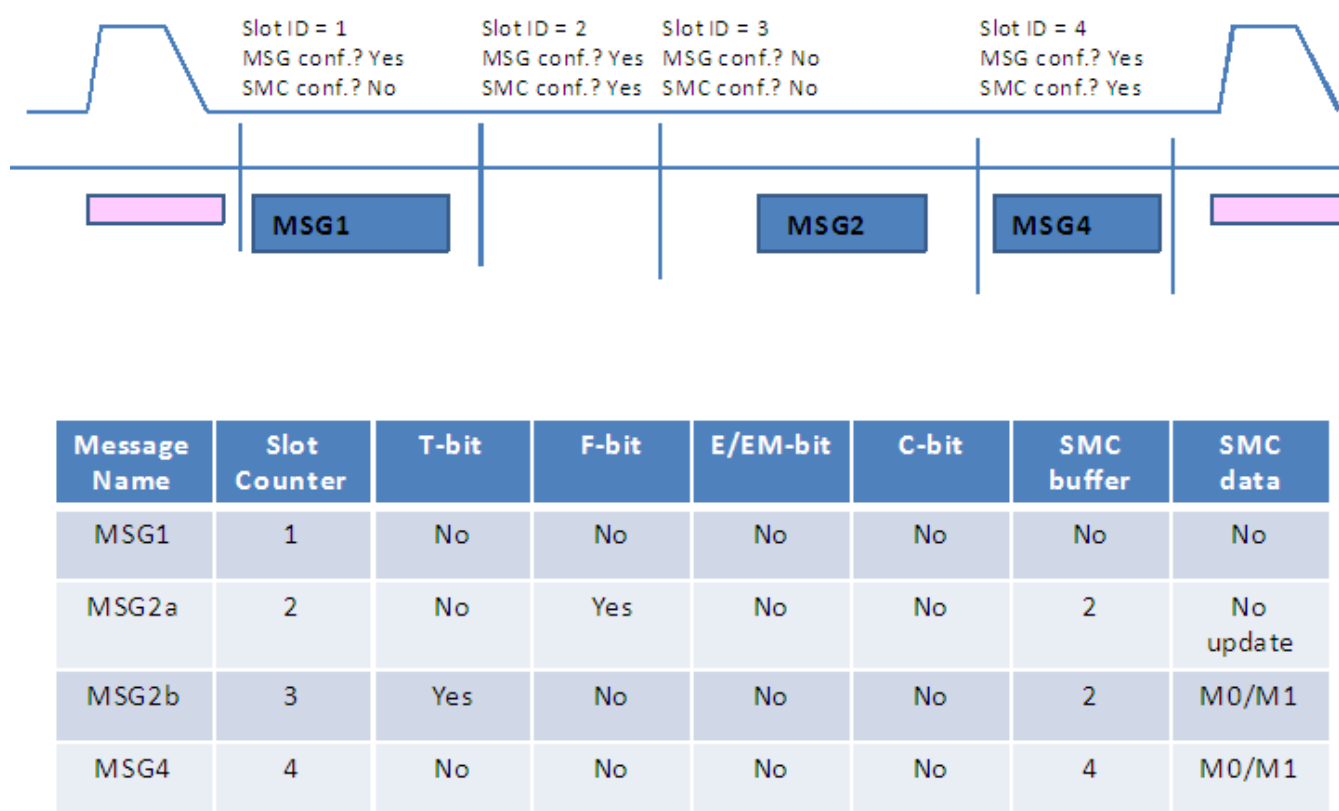


Figure 47-135. Message Much Too Late (in unconfigured slot)

In this scenario, MSG2 reception doesn't start in its own slot, and the F bit is set for message buffer location 2 (MSG2a).

Because slot 3 is not configured, the IP continues to wait for MSG2 in slot 3 and does not update the SMC message buffer 2. The IP recognizes the MSG2 start condition in the normal way and updates PSI5 message buffer 3 (MSG2b), capturing slot 3 as the start slot counter and setting T bit, so that the software can recognize that MSG2 has come in slot 3.

When MSG2 has been received, SMC message buffer 2 is updated at the end of slot 3. MSG4 is received normally.

NOTE

Any scenario where an expected message arrives in an unconfigured slot, leads to an issue when DMA access is used to read the PSI5 message buffer. This is because instead of the expected three entries in the PSI5 message buffer, there will be four, and assuming the DMA watermark is set for the expected three messages, the DMA request will be activated before MSG4 has been received.

In order to overcome this, if DMA access is required, no slots can be left unconfigured and the DMA watermark must be set to the number of slots in the SYNC period. In that way, the DMA will always be synchronised, but depending on how the 'empty' slot is configured, when a message is too late as described here, the result can be that the late message and its associated SMC content is lost.

47.5.2.3.4.6 Message Much Too Late (in unconfigured slot)

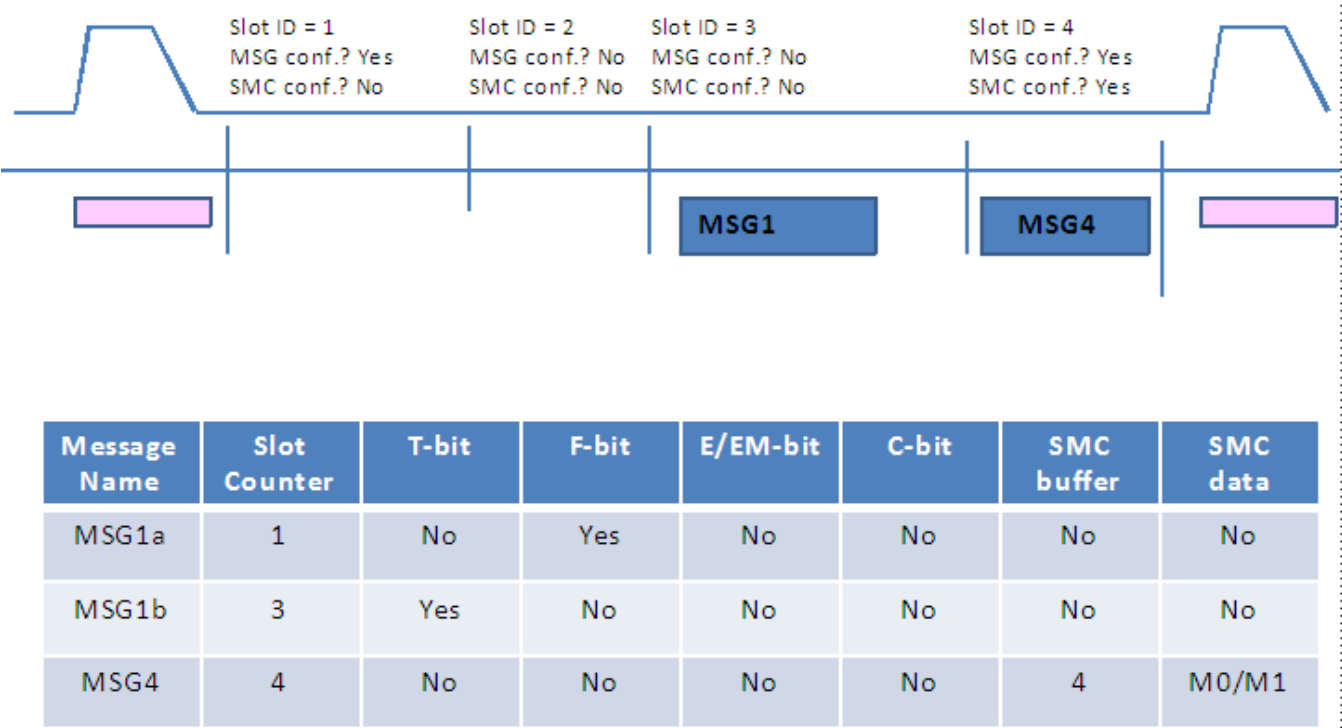


Figure 47-136. Message Much Too Late (in unconfigured slot)

In this scenario, MSG1 is received much too late, in the second of two unconfigured slots.

When MSG1 does not arrive in slot 1, MSG1a is saved in the message buffer with F bit set. The IP waits throughout the two unconfigured slots and is received in slot 3 with slot counter 3. Because MSG1b is received completely in slot 3 it is saved in the message buffer with T bit set.

MSG4 is received normally.

47.5.2.3.4.7 Message Too Late (in last slot)

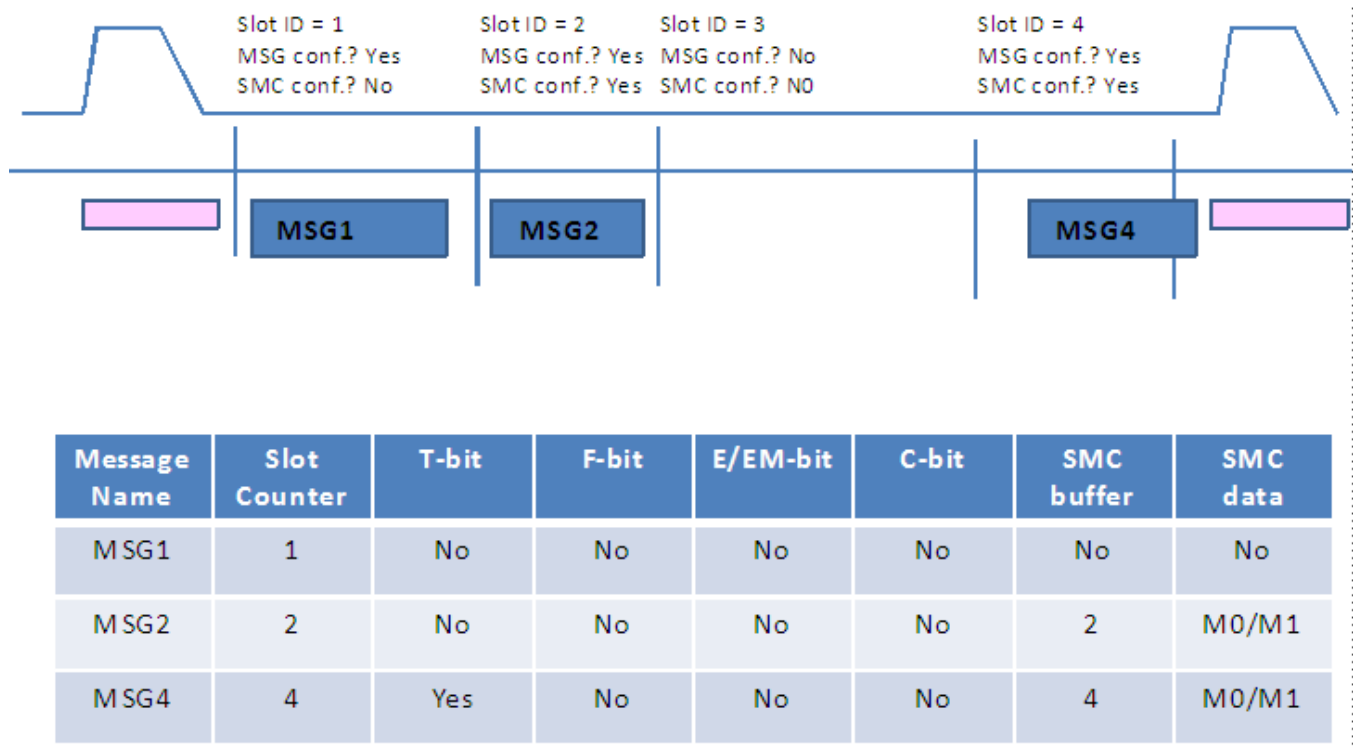


Figure 47-137. Message Much Too Late (in last slot)

In this scenario, the last message is received late.

MSG1 and MSG2 are received error free.

MSG4 is late with the start sequence identified in the correct slot 4, with slot counter 4 captured. The end of the message occurs after the end of the final slot, but before the Manchester decoder is disabled by the SYNC pulse on the bus. The message is received correctly with the T bit set.

47.5.2.3.4.8 Message Too Late (in last slot)

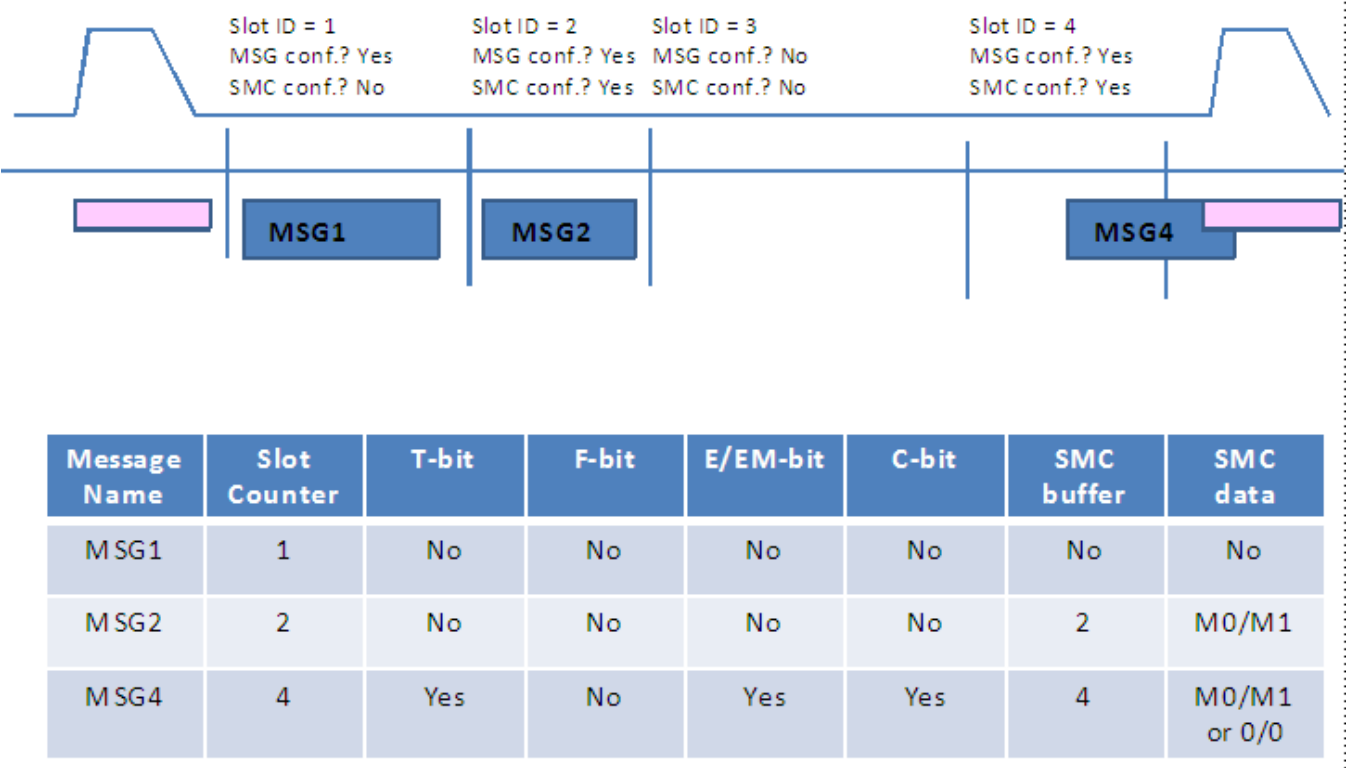


Figure 47-138. Message Much Too Late (in last slot)

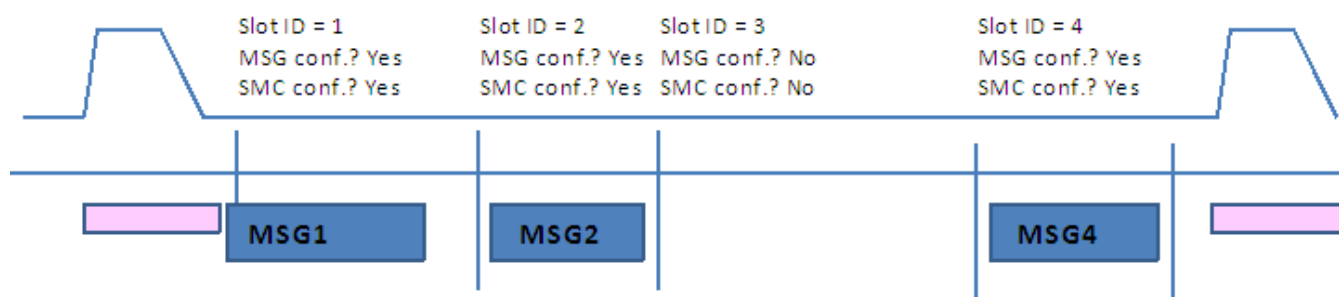
In this scenario, the last message is received late in the last slot.

MSG1 and MSG2 are received error free.

MSG4 is late with the start sequence identified in the correct slot 4, with slot counter 4 captured. The end of the message occurs after the Manchester decoder is disabled by the SYNC pulse on the bus, so the last part of the message cannot be received correctly. Nevertheless, the message reception is completed and the message is stored in the message buffer with the T bit set and 0 written in the buffer for all bits received while the Manchester decoder is disabled. This leads to E and C bit errors.

In an extreme case, it could also lead to EM bit errors, in which case the SMC buffer will also be written with (0,0) otherwise, the correct bits are written to the SMC buffer.

47.5.2.3.4.9 Message Too Early (in first slot)



Message Name	Slot Counter	T-bit	F-bit	E/EM-bit	C-bit	SMC buffer	SMC data
MSG1	0	Yes	No	No	No	1	M0/M1
MSG2	2	No	No	No	No	2	M0/M1
MSG4	4	No	No	No	No	4	M0/M1

Figure 47-139. Message Much Too Early (in first slot)

In this scenario, the first message is received early.

As soon as the Manchester decoder disable phase has ended, the IP verifies the idle condition on the signal line and then looks for the first message start sequence. MSG1 start sequence is received before the start of slot 1 with slot counter 0 captured. MSG1 is then received normally and saved with slot counter 0 and T bit set.

SMC bits M0/M1 are saved to SMC buffer 1.

MSG2 and MSG4 are received normally.

NOTE

This scenario can be avoided if the start of slot 1 is configured to coincide with the end of the Manchester decoder disable phase.

47.5.2.3.4.10 Message Too Early (in first slot and overlaps Manchester Decoder disable)

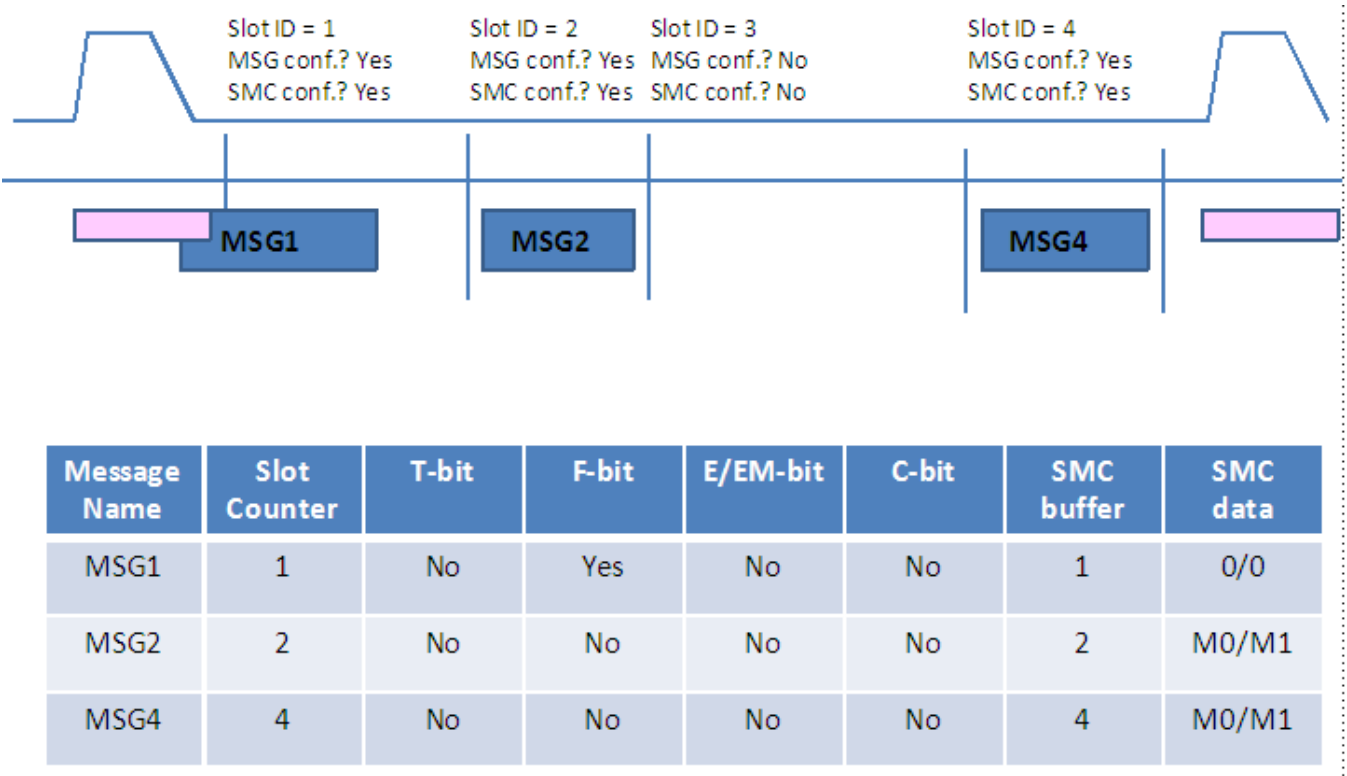


Figure 47-140. Message Too Early (in first slot)

In this scenario, the first message is received early and overlaps with the Manchester decoder disable phase.

As soon as the Manchester decoder disable phase has ended, the IP starts looking for “idle condition” on the signal line by monitoring it for 42 counts of the sample clock. If the signal line is found to be initially high or if a valid transition occurs in the time frame of 42 samples of the sample clock, the Manchester decoder stops decoding until the end of slot 1. Decoding will restart at the beginning of slot 2.

MSG1 is assumed to be lost and the “F” bit is set for that message with the Slot Counter as “1”. If MSG1 SMC is configured then SMC is saved with “00” appended for the M0/M1 bits.

47.5.2.3.4.11 Message Too Late (overlaps next message)

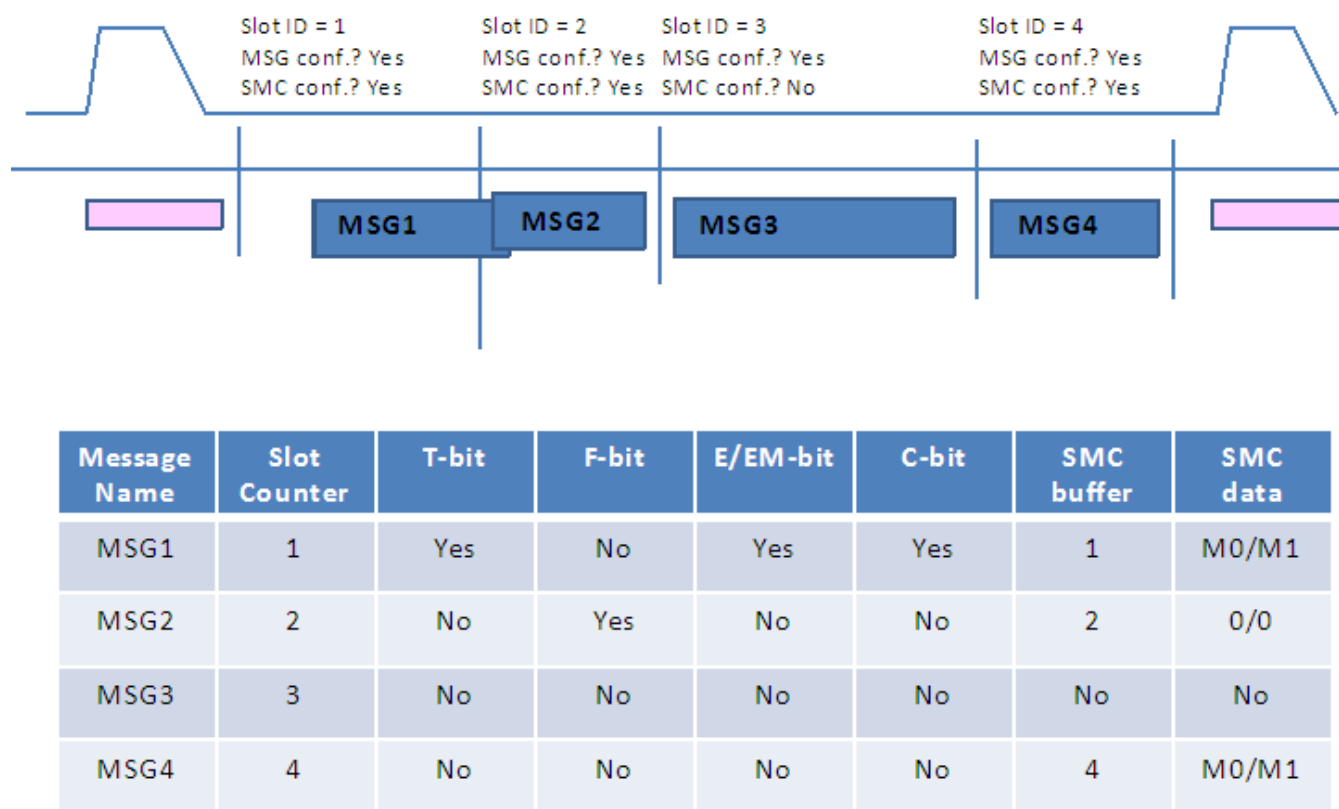


Figure 47-141. Message Too Late (overlaps next message)

In this scenario, MSG1 is received too late and overlaps with MSG2.

MSG1 start is recognized in slot 1. The last bits of MSG1 are disrupted by the start of MSG2 leading to likely E and C bit errors.

At the end of MSG1, idle time is found to be violated by either the signal line being high as soon as NO_DETECT_STATE is entered or FIRST_TRANSITION_DETECT occurs within 42 counts of the idle detect sample clock. The Manchester decoder stops decoding until the end of slot 2. Decoding is restarted at the start of slot3. MSG2 is not received and MSG2 location in the memory will be saved with the F-bit set. The SMC buffer 2 will be updated with 00.

MSG3 and MSG4 are received correctly in their own slots.

47.5.2.3.5 Data Frame Storage

Once the data frame is received and all required status fields updated (for example, Time stamp, C, E, EM, T, F and Frame Counter), this 64-bit value in the data receive register is moved to the RAM Register (PMB).

- This RAM buffer is a 32 (depth) × 64 (frame size) register array with DMA transfer capability (both single or burst).
- Also, the individual RAM locations can be accessed directly by the microcontroller core any time.
- Care needs to be taken that receive data is assigned to the sensor and frame slot unambiguously.

47.5.2.3.6 Message diagnostics

Apart from the regular status bits (mentioned above), a few more diagnostic checks are done and stored as status bits along with each data frame location in RAM. These are:

- Diagnosis registers are associated with each sensor data RAM buffer to indicate
 - RAM buffer has been filled with data.
 - RAM buffer has been overwritten before data previously stored was read.
 - Data written to buffer includes an error bit (E, EM, C, T, or F, configurable).
- The diagnosis registers are able to be read with a single DMA access.
- After successfully completing a read access, these registers are reset by hardware. This is done by mapping these diagnosis bits byte-wise so that one byte access clears a single bit without affecting other bits.

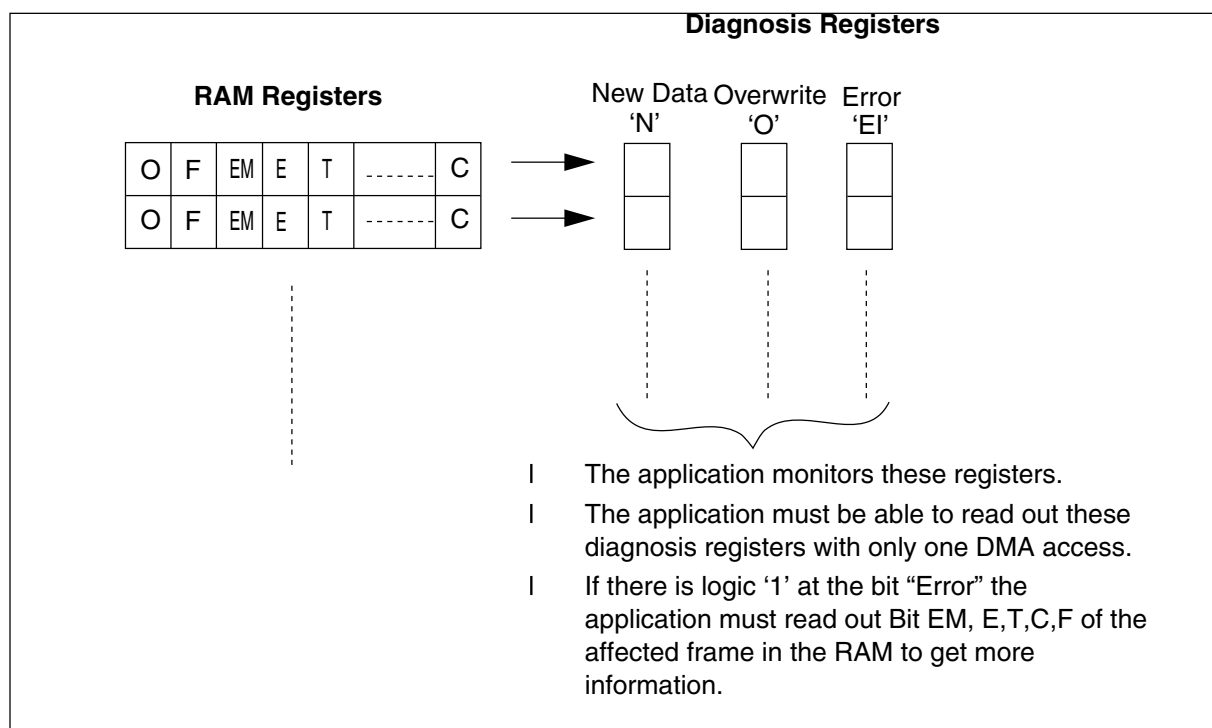


Figure 47-142. Diagnosis Information attached to every frame

Note

See [Implementation of EI Diagnostic Register](#) for more detail.

47.5.2.3.7 SMC message extraction

The optional messaging channel M0, M1 (shown in [Figure 47-127](#)) is used by sensors to send an SMC frame along with the normal PSI5 frame. It takes 18 PSI5 frames to send one SMC frame.

The extraction of SMC bits is done for each slot separately and stored separately into the SMC frame receive buffers. The size of the SMC frame receive buffers is 6 (no of slots) × 32 (depth of SMC frame, as defined below)

The extraction logic works as follows:

- Search for SMC start sequence 111111 followed by 0 on M1 and keep storing last 6 bits {CRC} on M0.
- If the start sequence is detected it continues with further extraction and storage of M1 and M0.
 - Store the extracted frame on the corresponding slot buffer.

- Move the SMC frame to corresponding SFR[i] register period. Once complete, the SMC frame is extracted and generates an interrupt, so that it can be read by the CPU.
- If the start sequence is not detected, it continues with the search.

The SMC frame is extracted in hardware and stored separately as memory-mapped registers. The size of the SMC frame receive buffers is 6 (no of slots) $\times$ 32 (depth of SMC frame).

47.5.2.4 Message Read Logic

PSI5 supports two kinds of read access to received sensor data.

- Read access through memory-mapped RAM registers. These registers can be read by the CPU any time or, specifically, when the concerned interrupts are raised. Applicable for both the SMC and the PSI5 messages.
- Read access through DMA request. Here the DMA sequencer treats the memory-mapped registers as a block and reads them sequentially. Note that even if the DMA is used, the CPU can read the RAM registers any time by addressing them. This is applicable for both the SMC and the PSI5 messages. When reading the PSI5 messages, it is possible to read only the PSI5 messages or PSI5 + diagnostic registers.

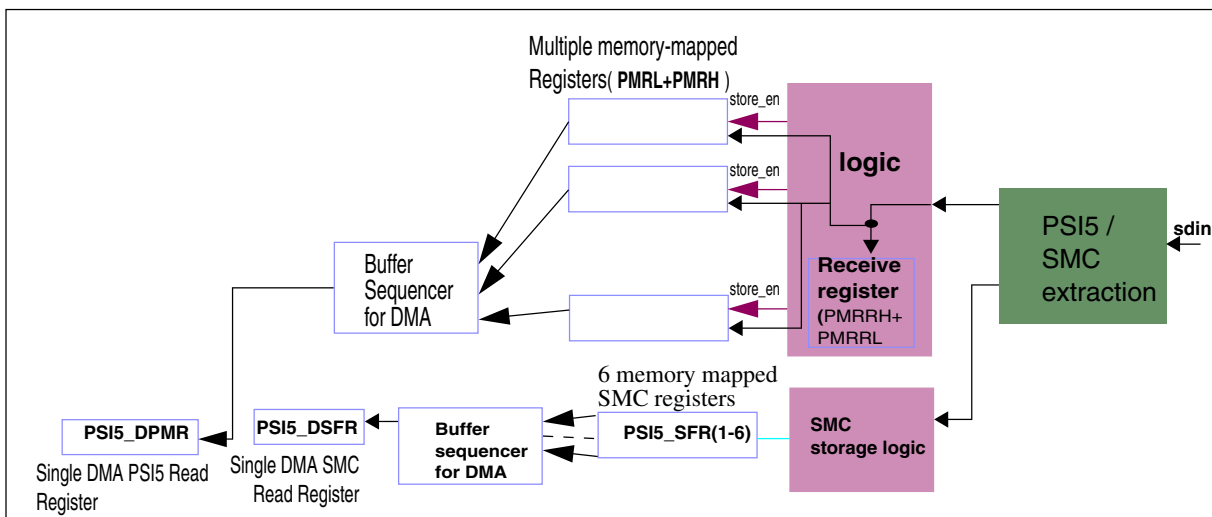


Figure 47-143. PSI5 and SMC Message Read Logic Block

47.5.2.5 CRC Recalculation

Error detection is realized by a single bit even parity (recommended for 10 or fewer bits) or a 3-bit CRC (recommended for long data words). The transmission error detection is selectable (by frame configuration register) as:

- 1-bit even parity
- 3-bit CRC

The generator polynomial of the CRC is $g(x)=1+x+x^3$ with a binary CRC initialization value of 111. The transmitter extends the data bits by three 0s (as MSBs). This augmented data word is fed (LSB first) into the shift registers of the CRC check. Start bits are ignored in this check. When the last zero of the augmentation is calculated on the input adder and the result is shifted, the buffer registers contain the CRC checksum. These three check bits are transmitted in reverse order (MSB first: C2, C1, C0).

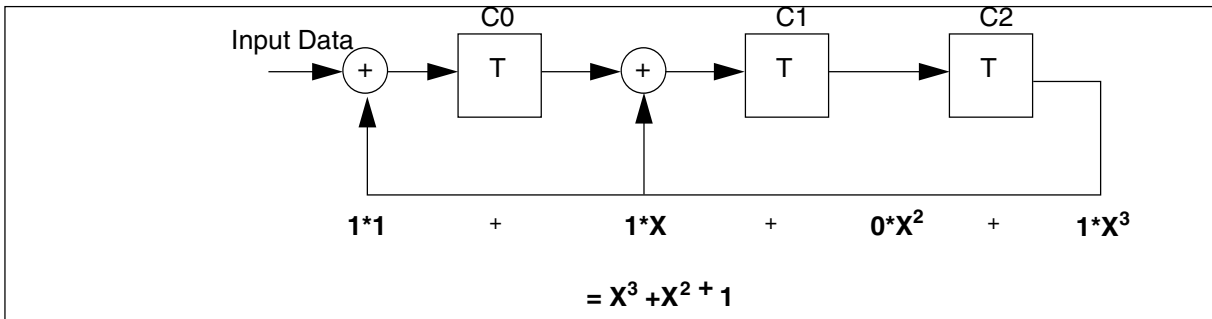


Figure 47-144. CRC calculation scheme

Table 47-134. CRC Calculation (at transmitter side)

Sensor Data (LSB first)		0	1	1	1	0	1	1	1	0	0	0	0	0
C0	1	1	0	1	1	0	0	0	0	1	1	1	0	0
C1	1	0	0	0	1	1	1	1	1	1	0	0	1	0
C2	1	1	0	0	0	1	1	1	1	1	1	0	0	1

Table 47-135. Transmitted and Received Sensor Data

Transmitted Sensor Data	S0	S1	D0	D1	D2	D3	D4	D5	D6	D7	D8	D9	C2	C1	C0
	0	0	0	1	1	1	0	1	1	1	0	0	1	0	0
Received Sensor Data	S0	S1	D0	D1	D2	D3	D4	D5	D6	D7	D8	D9	C2	C1	C0
	0	0	0	1	1	1	0	1	1	1	0	0	1	0	0

Table 47-136. CRC Re-Calculation (At Receiver Side)

Sensor Data (LSB first)		0	1	1	1	0	1	1	1	0	0	1	0	0
C0	1	1	0	1	1	0	0	0	0	1	1	0	0	0
C1	1	0	0	0	1	1	1	1	1	1	0	0	0	0
C2	1	1	0	0	0	1	1	1	1	1	1	0	0	0

Syndrome = 000 → Transmission successful

The correct transmission is checked by the PSI5 receiver by comparing the syndrome (content of the shift register after reception of both data bits and CRC bits) with the zero vector. For error free reception, all shift registers have to be zero.

47.5.3 ECU-to-sensor communication

Whereas the sensor-to-ECU communication is realized by current modulation, voltage modulation on the supply lines is used to transmit commands to the sensors. A predefined start sequence allows the sensors to recognize that a command is being transmitted during all synchronous operation modes.

47.5.3.1 Physical Layer

A logical 1/0 is represented by data sync pulse of variable length, a logical 1 by the sync pulse of width W1(programmable from 0 to 127 μ s) and logical '0' by sync pulse with W0{programmable from 0 to 127 μ s}.

47.5.3.2 Data Link Layer

47.5.3.3 Bit coding

The bit period is the cycle time as specified for the operation mode, for example, 500 μ s in the PSI5-P10P-500/3L mode. (For details about this topology, please refer [Section PSI5 topology](#)) The lower limit depends on the filter characteristics of the hardware circuitry used for the sync signal detection (subsequent sync signals might vary the internal $V_{ss, base}$ reference).

47.5.3.4 Data transmission

The IP supports the standard and the Non-Standard ECU to Sensor communication Formats . In the Non-Standard mode of communication it can be used to send any frame size programmable from 1-64 bits. In the Standard Formats it supports both the V1.3 and the V2.0 of the PSI5 ECU to Sensor Communication as briefed below in [Figure 47-145](#) and the text that follows it.

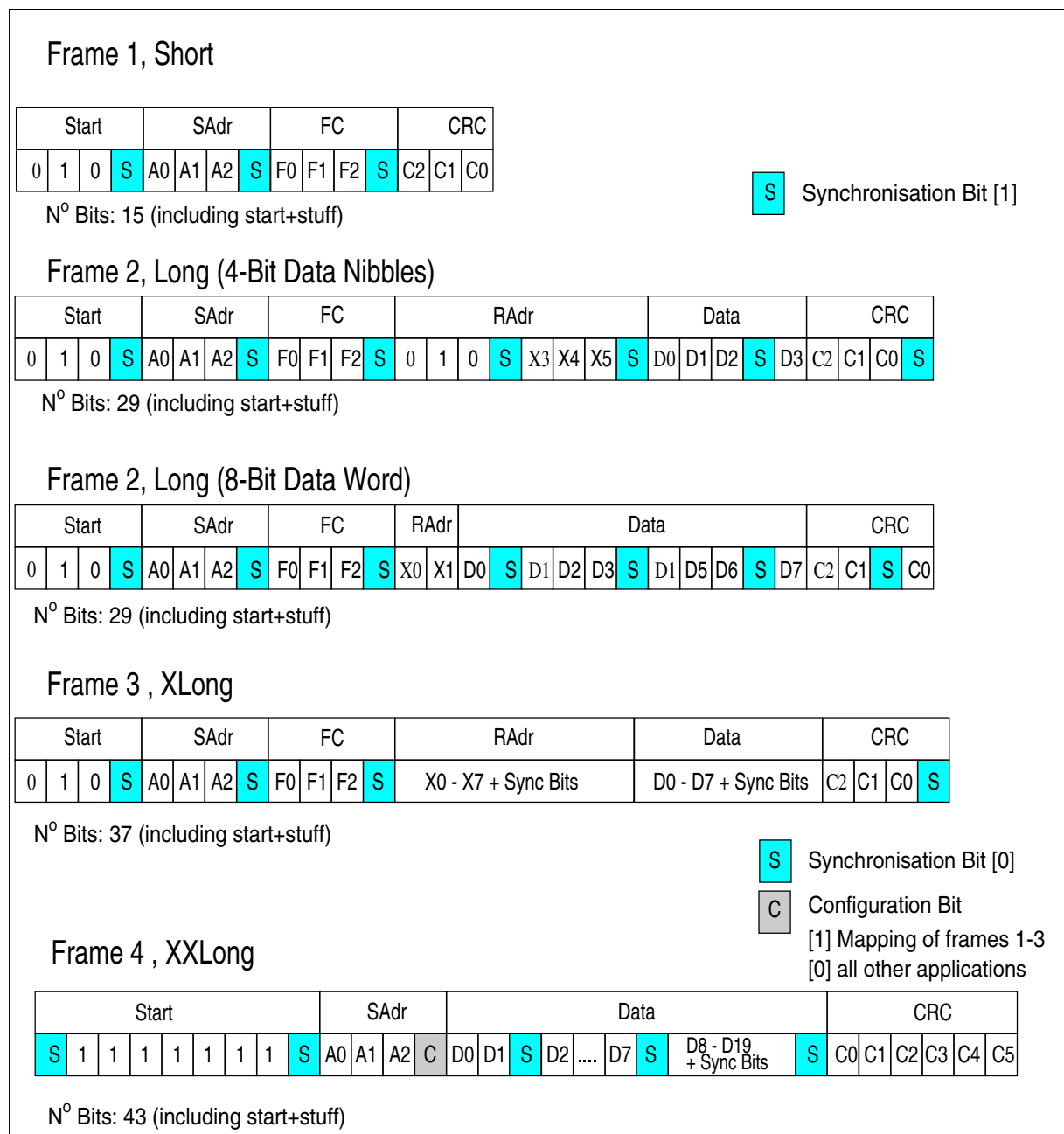


Figure 47-145. Standard Frame Fomats Supported

- **Short(V1.3)** : It consists of the following data stream (Order of transmission from IP is Left to Right. Thus C0 comes out Last) -- "010", S, "A0, A1, A2", S, "F0, F1, F2", S, "C2, C1, C0" where "010" is the start sequence "S" is the stuff bit(1), A0-A2 is the sensor address, F0-F2 are the functional codes and "C2-C0" is the CRC. Please refer [Figure 47-145](#) for the frame structure.
 - The CRC polynomial used is $x^3 + x^2 + 1$ with a binary initialisation value of "111(C2:C0)".
 - The transmitter extends the data bits by three zeros (as MSBs).
 - The CRC calculator is fed LSB first. Thus for CRC calculation the sequence given is [A0...A2,F0..F2,"000"] with "A0" being given first and "0" as the last. When the last zero of the augmentation is calculated on the input adder and the result is shifted, the CRC calculation has been achieved.
 - When mapping this sequence to the DPR register bits for calculating the CRC; then it becomes (DPR[0],DPR[1]....DPR[5],000) with DPR[0] being fed first in the CRC calculator.
- **Long(V1.3 - 4bit Data Nibble)** : It consists of the following data stream (Order of transmission from IP is Left to Right. Thus C0 comes out Last) -- "010", S, "A0, A1, A2", S, "F0, F1, F2", S, "X0, X1, X2", S, "X3, X4, X5, S, "D0, D1, D2", S,"D3, C2, C1", S, "C0" where "010" is the start sequence "S" is the stuff bit(1), A0-A2 is the sensor address, F0-F2 are the functional codes, "X0-X5" is the Range Address, "D0-D3" is the data and "C2-C0" is the CRC. Please refer [Figure 47-145](#) for the frame structure.
 - The CRC polynomial used is $x^3 + x^2 + 1$ with a binary initialisation value of "111(C2:C0)".
 - The transmitter extends the data bits by three zeros (as MSBs).
 - The CRC calculator is fed LSB first. Thus for CRC calculation the sequence given is [A0...A2,F0..F2,X0..X5,D0...D3,"000"] with "A0" being given first and "0" as the last. When the last zero of the augmentation is calculated on the input adder and the result is shifted, the CRC calculation has been achieved .
 - When mapping this sequence to the DPR register bits for calculating the CRC; then it becomes (DPR[0],DPR[1]....DPR[15],000) with DPR[0] being fed first in the CRC calculator.
- **Long(V1.3 - 8bit Data Word)** : It consists of the following data stream (Order of transmission from IP is Left to Right. Thus C0 comes out Last) Data Stream - "010", S, "A0, A1, A2", S, "F0, F1, F2", S, "X0, X1, D0", S, "D1, D2, D3, S, "D4, D4, D6", S,"D7, C2, C1", S, "C0" where "010" is the start sequence "S" is the stuff

bit(1), A0-A2 is the sensor address, F0-F2 are the functional codes, "X0-X1" is the Range Address, "D0-D7" is the data and "C2-C0" is the CRC .Please refer [Figure 47-145](#) for the frame structure.

- The CRC polynomial used is $x^3 + x^2 + 1$ with a binary initialisation value of "111(C2:C0)".
 - The transmitter extends the data bits by three zeros (as MSBs).
 - The CRC calculator is fed LSB first. Thus for CRC calculation the sequence given is [A0...A2,F0..F2,X0..X1,D0...D7,"000"] with "A0" being given first and "0" as the last. When the last zero of the augmentation is calculated on the input adder and the result is shifted, the CRC calculation has been achieved .
 - When mapping this sequence to the DPR register bits for calculating the CRC; then it becomes (DPR[0],DPR[1]....DPR[15],000) with DPR[0] being fed first in the CRC calculator.
- **X-Long(V1.3)** : It consists of the following data stream (Order of transmission from IP is Left to Right. Thus C0 comes out Last) Data Stream - "010", S, "A0, A1, A2", S, "F0, F1, F2", S, "X0, X1, X2", S, "X3, X4, X5", S, "X6, X7, D0", S, "D1, D2, D3", S, "D4, D5, D6", S, "D7, C2, C1", S, "C0" where "010" is the start sequence "S" is the stuff bit(1), A0-A2 is the sensor address, F0-F2 are the functional codes,"X0-X5" is the Range Address , "D0-D3" is the data and "C2-C0" is the CRC . Please refer [Figure 47-145](#) for the frame structure.
- The CRC polynomial used is $x^3 + x^2 + 1$ with a binary initialisation value of "111(C2:C0)".
 - The transmitter extends the data bits by three zeros (as MSBs).
 - The CRC calculator is fed LSB first. Thus for CRC calculation the sequence given is [A0...A2,F0..F2,X0...X7,D0...D7,"000"] with "A0" being given first and "0" as the last. When the last zero of the augmentation is calculated on the input adder and the result is shifted, the CRC calculation has been achieved .
 - When mapping this sequence to the DPR register bits for calculating the CRC; then it becomes (DPR[0],DPR[1]....DPR[21],000) with DPR[0] being fed first in the CRC calculator.
- **XX-Long(V2.0)** : It consists of the following data stream (Order of transmission from IP is Left to Right. Thus C5 comes Last) Data Stream - "011111110", "A0, A1, A2", "C", "D0, D1", S, "D2, D3, D4" D5, D6, D7, S, "D8, D9, D10, D11, D12, D13", S, "D14, D15, D16, D17, D18, D19", S, "C0, C1, C2, C3, C4, C5"

where "01111110" is the start sequence, "S" is the stuff bit(0), A0-A2 is the sensor address, "C" is the configuration bit, "D0-D19" is the data and "C0-C5" is the CRC. Please refer [Figure 47-145](#) for the frame structure.

- The CRC polynomial used is $x^6 + x^4 + x^3 + 1$ with a binary CRC initialization value 010101(C5:C0) .
- The transmitter extends the data bits by six zeros (as MSBs).
- The CRC calculator is fed LSB first. Thus for CRC calculation the sequence given is [A0...A2,D0..D19,"000000"] with "A0" being given first and "0" as the last. When the last zero of the augmentation is calculated on the input adder and the result is shifted, the CRC calculation has been achieved .
- When mapping this sequence to the DPR register bits for calculating the CRC ; then it becomes (DPR[0],DPR[1]....DPR[23],000000) with DPR[0] being fed first in the CRC calculator.

47.5.3.4.1 Transmission Data Framing

The output data transmission block is constructed of seven sub-blocks:

- Data preparation register
- Buffer register
- Data shift register
- Periodic Sync Pulse Trigger Generator
- "GTM driven" sync pulse trigger input (GTM external to PSI5)
- Pulse length modulation block
- OR gate

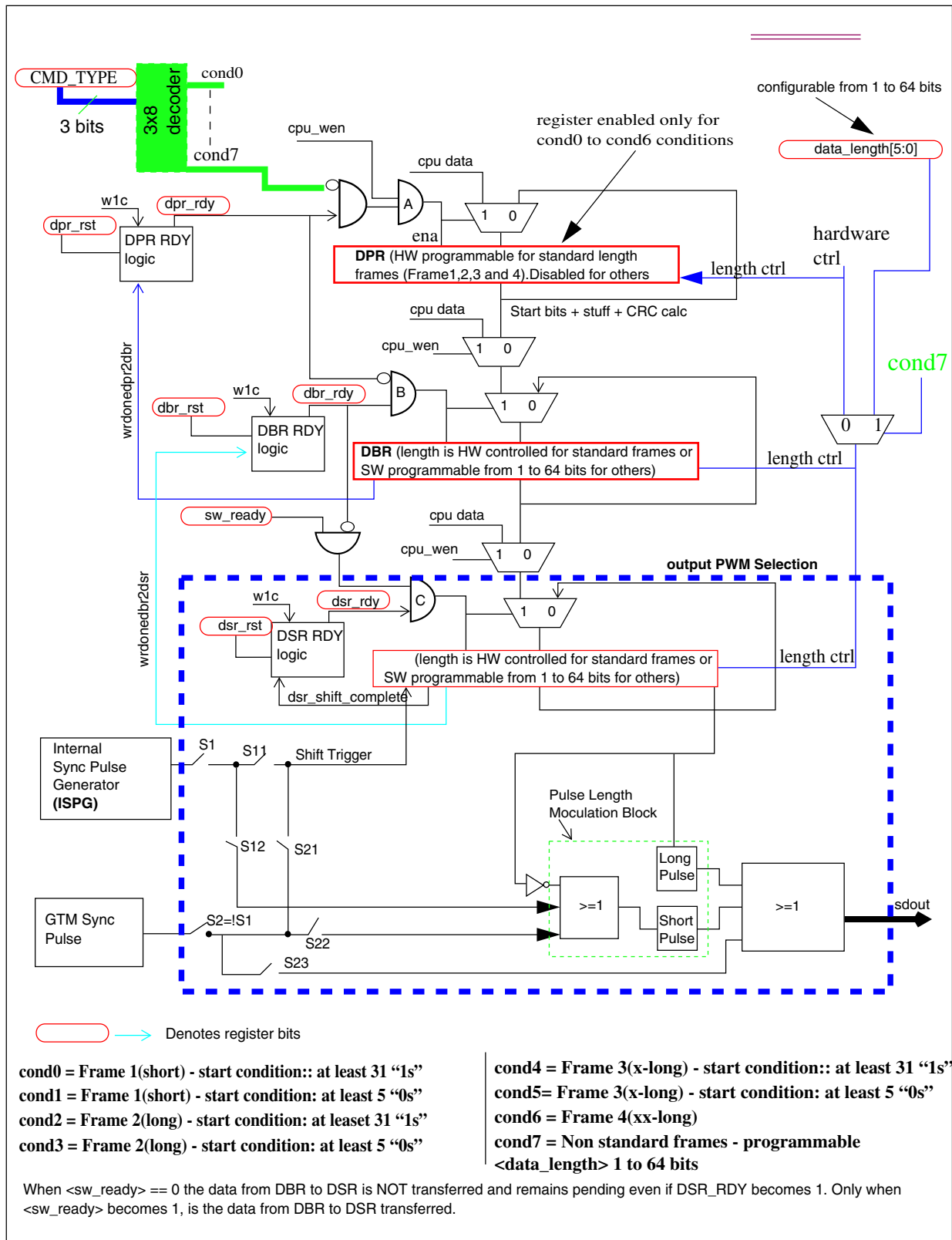


Figure 47-146. Data Transmission Block

47.5.3.4.2 Data Preparation Register (64-bit register, DPRH + DPRL, DPRH is fixed at 0)

Upto 24 bits of this register can be written by the CPU. These register bits contain the ECU to sensor command that needs to be transmitted after being appended with the CRC, start bits and stuff bits. The hardware automatically calculates the CRC and adds the start and the stuff bits, before transferring the command to DBR.

- Having written the data to the DPR, the DPR_RDY bit has be cleared with a w1c. This indicates to the internal logic that the data in the DPR is valid and is to be processed.
- If DPR_RDY = 0 and a new command is written to the DPR then the command is rejected and GISR[IS_PROW] is set.
- If the CRC or parity bit calculation is finished, then the data is shifted to the Buffer register with the start bits and the sync bits appended as well. This transfer is done to the Buffer register only if the Buffer register is ready for new data, which is indicated by GISR[DBR_RDY] == 1.
- As mentioned above, once this tranfer is done, the DPR_RDY becomes 1 and a new command can be written to it.

47.5.3.4.3 Data Buffer Register (64-bit register)

- This register has two subregisters DBRH (higher register) and DBRL (lower register). Both of them are collectively referred to as DBR. DBR is of length 64 bits DBR[63:0]
- The bits accessible for functionality are dependent on the value of PSI5_DOBCR[CMD_TYPE]. For the writeable bits corresponding to different setting of the PSI5_DOBCR[CMD_TYPE] field please refer bit description in CMD_TYPE[2:0]
- When DPR_RDY == 0 and DBR_RDY goes to 1 then internal logic automatically transfers the data from the DPR (with CRC + stuff bits + sync bits) to the DBR. DBR_RDY goes low and DPR_RDY goes 1.
- If DBR_RDY == 1 AND DPR_RDY == 1, then the data to be transmitted to sensors can be written to this register directly via system bus skipping preparation register in case application includes the start and sync bits and calculates the CRC/Parity itself. If the CPU writes to the DBR when DBR_RDY == 1 but DPR_RDY == 0 and PSI5_DOBCR[CMD_TYPE] != 7, then the results can be unpredictable since the DPR has a pending data that needs to be transferred to DBR.

- If the CPU is writing to the DBR as mentioned above, then the DBR_RDY bit has to be cleared by a w1c. This indicates to the internal logic that the data in the DBR is valid and is to be used for the shift operation.
- If while DBR_RDY == 0, and a new command is directly written to the DBR then the command is rejected and GISR[IS_BROW] bit is set.
- If while DBR_RDY == 0 a new command HAS to be written in the DBR then DOBCR[DBR_RST] has to be written to 1. Writing a 1 to DBR_RST will reject the current command in the DBR and generate DBR_RDY = 1.
- If the Data Shift Register signals that it is ready for new data (PSI5_GISR[DSR_RDY] == 1), the data of the Buffer register is shifted into the data shift register provided PSI5_DOBCR[SW_READY] is 1.
- If PSI5_DOBCR[SW_READY] == 0 then the data from the DBR is NOT transferred to the DSR automatically even if the DSR_RDY goes as 1. In such a case the IP waits for the software to make the PSI5_DOBCR[SW_READY] bit as 1; only then does it transfer the contents to the DSR.

47.5.3.4.4 Data Shift Register (64-bit register)

- This register has two subregisters DSRH (higher register) and DSRL (lower register). Both of them are collectively referred to as DSR. DSR is of length 64 bits DSR[63:0]. This register has a 1-1 correspondence with the bits in the DBR register when this register is updated automatically by the hardware.
- The bits accessible for functionality are dependent on the value of PSI5_DOBCR[CMD_TYPE]. For the writeable bits corresponding to different setting of the PSI5_DOBCR[CMD_TYPE] field please refer bit description in CMD_TYPE[2:0].
- When DBR_RDY == 0 and DSR_RDY goes to 1 then internal logic automatically transfers the data from the DBR to the DSR provided PSI5_DOBCR[SW_READY] == 1'b1, else this transfer remains pending till PSI5_DOBCR[SW_READY] is '0'. Once the transfer is complete, DSR_RDY goes low and DBR_RDY goes 1. DSR contents that would finally get transferred, can be read at this stage, i.e., when DBR_RDY becomes "1".
- Making PSI5_DOBCR[SW_READY] now as '0' will not interrupt the ongoing transfer in the DSR.

- If `DPR_RDY == 1 AND DBR_RDY == 1 AND DSR_RDY == 1`, then the data to be transmitted to sensors can be written to this register directly via system bus skipping Buffer register. If the CPU writes to the DSR when `DSR_RDY == 1` but `DBR_RDY == 0` then the results can be unpredictable since the DBR has a pending data that needs to be transferred to DSR.
- When `PSI5_DOBCCR[CMD_TYPE] == 7` and If `DBR_RDY == 1 AND DSR_RDY == 1`, then the data to be transmitted to sensors can be written to this register directly via system bus skipping Buffer register. If the CPU writes to the DSR when `DSR_RDY == 1` but `DBR_RDY == 0` then the results can be unpredictable since the DBR has a pending data that needs to be transferred to DSR.
- If the CPU is writing to the DSR as mentioned above then the `DSR_RDY` bit has to be cleared by a `wlc`. This indicates to the internal logic that the data in the DSR is valid and is to be used for the shift operation.
- If while `DSR_RDY == 0`, and a new command is directly written to the DSR then the command is rejected and `GISR[IS_DSROW]` is set.
- If while `DSR_RDY == 0` a new command HAS to be written in the DSR then `DOBCCR[DSR_RST]` has to be written to 1. Writing a 1 to `DSR_RST` will reject the current command in the DSR and generate `DSR_RDY = 1`.
- This register is clocked by the ISPG pulses or the GTM pulses, as described in [Sync Pulse Trigger Generation](#).
- Once the shift operation is complete in the DSR then `DSR_RDY` becomes 1.

47.5.3.4.5 Sync Pulse Trigger Generation

The sync pulse trigger can be used to generate the timing pulses or the data pulses.

The sync pulse can be generated based on triggers from two different sources:

- **ISPG (Internal Sync Pulse Generator):** Composed of CTPR, CIPR and Channel Target Counter (per channel). The internal sync pulse generator is clocked by the internal 1 MHz clock. The Channel Target Counter operates with a 1 μ s resolution (when using the 1 MHz clock). The ISPG is implemented to allow generation of internal periodic trigger. Please Refer [Figure 47-151](#) for the ISPG structure and [Figure 47-150](#) for the CTPR and CIPR explanation. The ISPG uses the Channel Target Counters (CTCs) for generating the sync pulse. The CTCs can be triggered globally (simultaneous for all channels) or locally (individual to each channel). The `CTC_GED` bit in the Global Control Register (GCR) ensures that all the CTCs start/stop simultaneously. The individual target values when the trigger pulse is generated are set in the Counter Target Pulse Register (CTPR) and the Channel Initial Pulse

Register (CIPR). These registers are present for each channel. When staggering the channels, the CTC_GED bit should be used for providing a simultaneous trigger to the staggered channels. It is also possible for a subset of channels to work on the above arrangement while the remaining channels independently generate trigger pulses for which start/stop is controlled locally to the channels.

- For each channel, it is possible to configure the trigger period by software using sync pulse generator control/configuration register and setting compare values in Counter Target Pulse Register(s). The Counter Initial Pulse Register (CIPR) can be used to provide the initial offset between various channels.
- It is possible to deactivate each channel separately for the purpose of Sync Pulse generation.
- The periodic pulses can be used to synchronize sensor-to-ECU communication (synchronization timing pulses) [State 1 of [Valid states for integrated sync pulse generator](#)] or can be used as data pulse during ECU-to-sensor communication (through the DOBCR) [State 2 of [Valid states for integrated sync pulse generator](#)].
- When programming the CTPR, the Tsync period should be programmed ≥ 6 SP_TS_CLK cycles (For details of SP_TS_CLK, refer to Bit 23 of Section 54.3.2.2, PSI5 Channel Control Register (PSI5_PCCR))
- **GTM event controlled:** Generation of sync Trigger/Pulse can be done based on events (e.g., crank angle or software) through **general timer module (GTM)** output to the PSI5 interface.
 - Here the sync trigger pulses come from an external GTM module.
 - The sync pulse programming (period, etc.) is done inside GTM.
 - In [Figure 47-146](#), if switches S2 and S21 are closed and S22 and S23 are open, the GTM trigger is used for data pulse generation (data pulses for ECU-to-sensor communication). [State 4 of [Valid states for integrated sync pulse generator](#)]

- In [Figure 47-146](#), if switches S2 and S22 are closed and S21 and S23 are open, the GTM trigger is used for timing pulse generation (synchronization pulse for sensor-to-ECU communication). [State 3 of [Valid states for integrated sync pulse generator](#)]
- In [Figure 47-146](#), if switches S2 and S23 are closed and S21 and S22 are open, the GTM trigger is passed directly to data sync port without pulse length modulation. In this case the sync pulse length is defined by the GTM pulse length. [State 5 of [Valid states for integrated sync pulse generator](#)]
- **Programming and control :** For achieving the various states of ISPG control or the GTM control, the switches S1, S2, S11, S12, S21, S22, and S23, shown in [Figure 47-146](#) can be set (closed) or cleared (open) by writing to the three equivalent bits (GTM_TRIG_SEL, SPULSE_SEL and OP_SEL) belonging to the DOBCR, as shown in [Figure 47-147](#). The programming of various states mentioned above, with reference to the [Valid states for integrated sync pulse generator](#) can be seen in [Table 47-4](#) and also shown in [Figure 47-153](#). The sending of the ECU to sensor commands, can be controlled by PSI5_GISR[DPR_RDY, DBR_RDY, DSR_RDY] bits and the PSI5_DOBCR register bits.

47.5.3.4.5.1 Timing Pulses

Timing pulses can also be produced without pulse width modulation also . These timing pulses can be generated through two sources:

- Periodic sync pulse from ISPG . In this case the synchro pulse from the internal sync pulse generator can be made available without any modulation. Referring [Figure 47-147](#), in synchronous mode , if DOBCR[GTM_TRIG_SEL] == 0 and DOBCR[OP_SEL] == 0 then the internal sync pulse generator output can be made available without any modulation . The high time of this pulse is fixed at 4 μ s.
- Referring [Figure 47-147](#), in synchronous mode , if DOBCR[GTM_TRIG_SEL] == 1 and DOBCR[OP_SEL] == 0 , then the GTM trigger can be used for timing pulse generation without any Pulse Width Modulation.

47.5.3.4.5.2 Data pulse generation

Data pulses are produced through Pulse Length modulation block based on triggers from one of the two selectable sources **ISPG** or the **GTM**, during ECU-to-sensor communication. The selection between the above two sync sources can be done by software programming Pulse length control/configuration register.

- A logical '1' in Data Output Register LSB is coded as data sync pulse of width W1 {up to 127 μ s}.
- A logical '0' in Data Output Register LSB is coded as data sync pulse of width W0 {up to 127 μ s}.
- The two widths are programmable in Pulse Length Control/Config Register.
- The Data Shift Register is shifted Right by everytime as bit is transmitted.

Figure 47-146 shows the data transmission block.

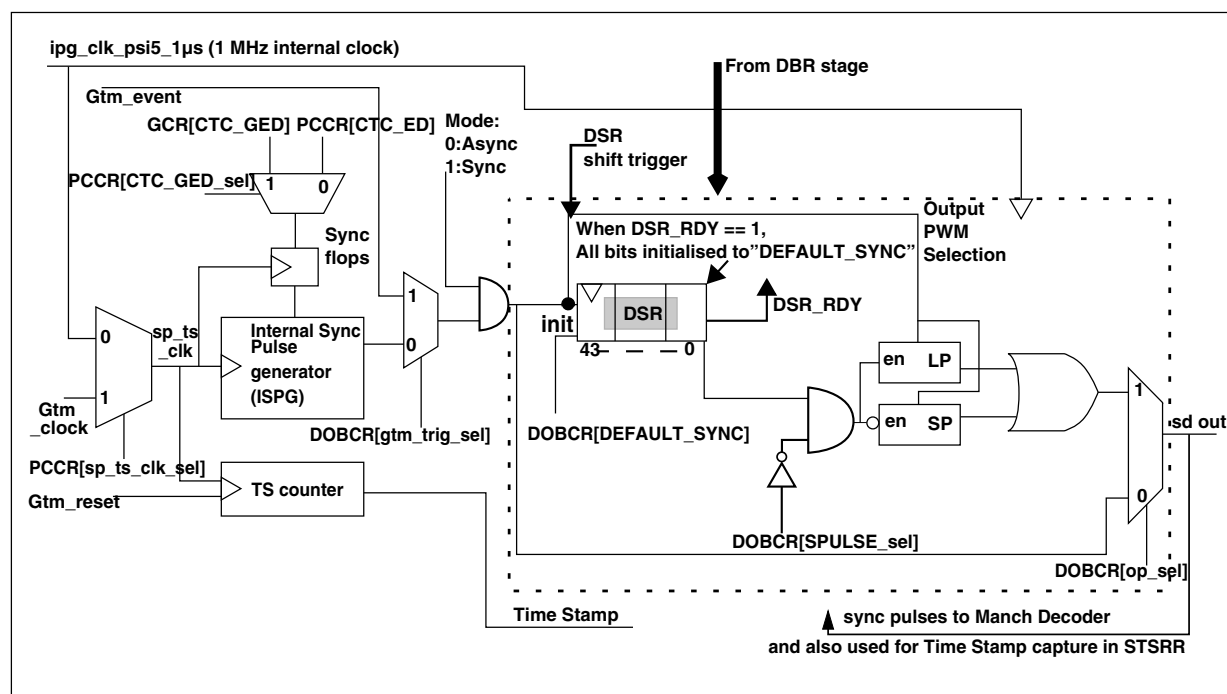


Figure 47-147. ECU-to-sensor communication registers (implementation of the Data Transmission Block figure)

The above figure details the implementation of Figure 47-146 and shows how the various bits mentioned in Section 54.3.2.30, Data Output Block Configuration Register (PSI5_DOBCR) control the behavior of the Output Command path. The above figure also shows the various clock sources to the Internal Sync Pulse Generator and the Time Stamp counter.

47.5.3.5 Response to ECU-to-sensor frames

The response to an ECU-to-sensor frame can be optionally transmitted in two ways:

- Out-of range values of data frame (as fast response coded as normal PSI5 payload region)
- Serial messaging channel (verification of the response frames must be handled by the software)

47.5.4 Tx/Rx scenario

The diagram below shows the communication scenario in synchronous modes. For asynchronous mode, sync pulses are not generated and sensor transmits data asynchronously.

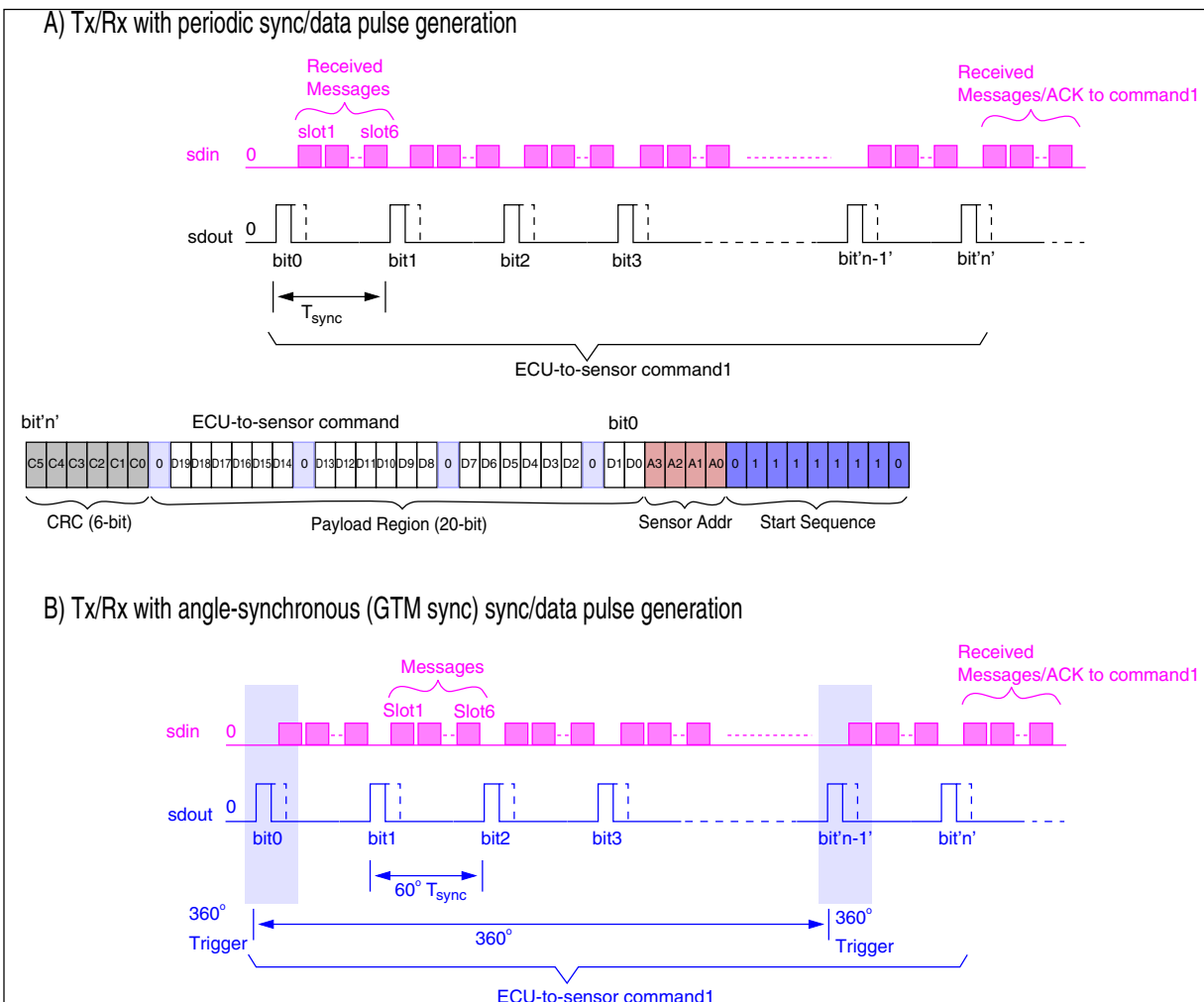


Figure 47-148. Synchronous mode communication scenarios

47.5.5 Interrupt handling

Configurable interrupts based on events (for example, reception of valid message), errors, DMA transfers, and so on are implemented. The following types of events can generate an interrupt:

- Reception of a new PSI5 message
- Reception of a new SMC message
- Unread SMC message overwritten in the SMC buffer area (PSI5_SFR)
- Unread PSI5 message overwritten in the PSI5 buffer area (PSI5_PMRH, PSI5_PMRL)
- Dedicated interrupts for selectable errors, mentioned below:
 - Bit C is set
 - Bit E and/or EM is set
 - Bit T is set
 - No PSI5 frame was received in a defined frame slot (Bit F is set)
- ECU-to-sensor communication-related interrupts
 - The application tries to load new data into the data preparation register before the register has signaled that it is ready for new data. (Note that the new command will be ignored in this case.)
 - The application loads new data into the data buffer register before the register has signaled that it is ready for new data. (Note that the new command will be ignored in this case.)
 - The application loads new data into the data shift register before the register has signaled that it is ready for new data. (Note that the new command will be ignored in this case.)
- DMA related interrupts (transfer finished/FIFO_FULL/underflow)

Various interrupt signals that represent the above events are:

- PSI5 new message arrival interrupt (**Status flags in PSI5_NDSR**) This is asserted when any of the programmed storage PSI5 locations (max 32) receive a new PSI5 message (error/error free). This interrupt also indicates that the PSI5 receive register (PMRRL/PMRRH) has been updated with a new PSI5 message.

- SMC message, ECU-to-sensor communication, and Time Stamp related interrupts (**All Status flags in PSI5_GISR**) : This is asserted during following events:
 - When a new SMC message is received in its corresponding slot OR
 - When there is a CRC error in any SMC message OR
 - When an unread message in some corresponding SMC location (in PSI5_SFR) gets overwritten by a new SMC message OR
 - When DPR is ready for receiving a new data OR
 - When DBR is ready for receiving a new data OR
 - When DSR is ready for receiving a new data OR
 - When there is an attempt to write to DPR while it is NOT ready OR
 - When there is an attempt to write to DBR while it is NOT ready OR
 - When there is an attempt to write to DSR while it is NOT ready OR
 - When data time stamp register is updated with new time stamp value OR
 - When the sync time stamp read register is updated with a new time stamp.
- PSI5 Message Overwrite Interrupt (**Status Flags in PSI5_OWSR**) : This is asserted when any of the programmed storage locations in the memory (max 32) have an unread PSI5 message that is overwritten by a new upcoming PSI5 message. Corresponds to OWSR.
- PSI5 Error interrupt (**Status Flags in PSI5_EISR**) : This is asserted when any of the programmed storage locations in the memory (max32) have an error in the PSI5 message. It is used only for PSI5 messages.
- DMA Interrupt (**Status Flags in PSI5_DSR**) :
 - When the DMA transfer corresponding to conf2, conf3 or conf4 finishes OR
 - When the DMA transfer corresponding to SMC message gets over OR
 - When there is an FIFO_FULL/Underflow in the PSI5 RAM register area OR
 - When there is an underflow in the SMC message area.

47.5.5.1 Error conditions

- C: CRC/parity error on received data, evaluated after recalculating CRC on received data and comparing with received CRC or recalculating parity on received data and comparing with received parity.
- E: Electrical error during any Tbit period (excluding Messaging field M0M1) during frame reception (not applicable during Idle state); that is, if at least one bit is absent during its Tbit period.
- EM: Electrical error on M0M1 field.
- T: Timing error during frame reception violates the slot boundary. Here, all the corner cases and conditions are taken care of (for example, the isolation of correct slot to correct message if the message is too early or too late with respect to its expected position, the crossing of the slot boundary when one frame overlaps the other, and so on).
- F: No frame received during a particular slot time in case of sync mode.

47.5.6 DMA support

NOTE

DMA master ID replication mode is not allowed together with PSI5.

- The PSI5 module generates separate DMA transfer requests for PSI5 message/diagnostic status and SMC messages.
- A single DMA request is used for PSI5 messages and diagnostic data. The request is triggered based on the selected DMA configuration (either PSI5 message or Diagnostic data or both). After choosing the configuration, data is available on respective DMA pop-up registers.
- DMA requests for PSI5 messages and diagnostic registers are based on the watermark level of message buffers (to be treated as FIFO for the DMA). PSI5 message buffer FIFOs get data from the PSI5 receiver in the order they are received.

- DMA requests for SMC messages are based on successful storage of each SMC frame to the message buffers. The SMC message buffers get data from SMC extraction logic in fixed round-robin arbitration sequence.
- Please see [Single DMA request for PSI5/diagnostic message buffers](#) for PSI5 DMA handling and [Single DMA request for SMC received messages](#) for SMC DMA handling

47.5.6.1 Single DMA request for PSI5/diagnostic message buffers

A common DMA request (per PSI5 block) is generated for PSI5 message/diagnostic buffers. The depth (up to 32 in present context) of the message buffer is programmable via the MEM_DEPTH filled in the PCCR. The width of each location of the PSI5 message buffer is 64 bits. When reading PSI5 message through the DMA, the 32-bit DPMR is sequenced with the data from the PSI5 message buffer. When reading the diagnostic registers through the DMA, the 32-bit DDSR is sequenced as NDSR followed by EISR. The DMA operation will happen as follows:

- Based on the DMA configuration chosen, one of the three DMA transfer sequence is initiated:
 - PSI5 buffer followed by diagnostic data sequenced to DPMR
 - PSI5 data sequenced to DPMR
 - Diagnostic buffer data sequenced to DDSR
- The DMA request is asserted when the number of messages stored in the buffer crosses the configured watermark number of messages.
- DMA request is deasserted once the "watermark" number of messages have been read.
- If the number of unread messages is still greater than the watermark programmed, then the DMA request is reasserted.
- The received message data is read from a common memory-mapped 32-bit DMA PSI5 message register (DPMR). Thus, a single PSI5 message (64 bits wide) will require two IPS read operations. The diagnostic data is read either through the DPMR or through the DDSR (single 32-bit register) by the DMA controller via the IPS interface.
- It is assumed that the DMA Controller's Transfer Control Descriptor (TCD) would be set to transfer bytes equal to the messages indicated by watermark level, as described in DMA_PM_DS_WM[4:0] in [DMA Control Register \(PSI5_DCR\)](#). It is assumed

that when responding to a DMA request, the DMA controller will read faster than the rate at which messages are received and stored in the buffer/FIFO which will cause the DMA request to deassert.

- The next received message will be updated on the common memory map register once the previous message is read by the DMA (by the DMA sequencer).
- The store signal from the synchronization logic will push the message into the memory-mapped RAM registers and in parallel also into the PSI5 receive register. [Figure 47-143](#) shows the format in which the messages will be stored in any message buffer (RAM registers) and finally when being read from DMA access register.
- For more details about DMA programming, please refer to [PSI5 Channel Control Register \(PSI5_PCCR\)](#), [DMA Control Register \(PSI5_DCR\)](#), [DMA Status Register \(PSI5_DSR\)](#), [DMA PSI5 Message Register \(PSI5_DPMR\)](#) and [DMA Diagnostic Status Register \(PSI5_DDSR\)](#).

47.5.6.2 Single DMA request for SMC received messages

A separate DMA request (each channel) is generated for the SMC message buffer. Each SMC received messages would be stored in the internal registers and read out by DMA controller via a common memory-mapped DMA SMC Frame Register (DSFR) (single 32-bit register for each message buffer). The data flow will happen as follows:

- A request is asserted when the system buffer successfully receives one complete SMC message, and remains asserted until all complete messages have been read. Data is read from the DMA PSI5 Message Register in the memory map.
- It is the job of an internal sequencing logic to provide the next complete received message. The internal sequencer works in a round-robin fashion checking for the slot-specific registers for each SMC message. Thus the sequence will be slot1--slot2-- slot3-- slot4--slot5--slot6--slot1 and so on. In whichever slot it encounters the first complete SMC message the DMA request would be asserted and the sequencer will wait for this data to be read before moving on to the next slot in its loop. If a particular slot message remains unread and a new SMC message for the same slot comes in then the previous message would be overwritten and the OW bit would be set.

- The DMA controller should be programmed to read only one message in one DMA transfer. Otherwise, if it is programmed to read more than one message in one DMA transfer, underflow in buffers might occur. User software should ensure there is no overflow.
- For more details about DMA programming please refer "PCCR-PSI5 Channel Control Register" on page 267, "DCR-DMA Control Register" on page 272, "DSR-DMA Status Register" on page 276 and "DSFR-DMA SMC Frame Register" on page 284

47.5.7 Message read through interrupt generation

There is a separate interrupt request for received messages for each PSI5 channel. Each PSI5 channel can be configured individually to generate an interrupt on reception of messages. An interrupt is asserted when the corresponding PSI5 channel has received a complete message (PSI5 or SMC). Please see [Interrupt handling](#) for more details about interrupts. The interrupt routine should be coded in such a way to take the least amount of overhead (in CPU cycles) while entering the routine and start reading the available messages. The routine should be capable of accessing the corresponding PSI5 channel which is enabled to give out messages via interrupts through the IPS Interface. In response to these interrupts, the CPU is expected to read interrupt status register for individual PSI5 Channel (depending on which interrupt was asserted) and store it in system memory.

47.6 Initialization information

47.6.1 Initialization

47.6.1.1 Sensor startup and initialization

After each power-on or under-voltage reset, the sensor performs an internal initialization which is divided into three phases:

- Initialization Phase1: No data transmission.

- Initialization Phase 2: Transmission of type code and serial number of sensor.
- Initialization Phase 3: Transmission of "Sensor Ready" or "Sensor Defect" or other sensor specific data.

As this PSI5 implementation is targeted towards powertrain devices/sensors the standard initialization phase is modified and tuned for those applications. The comparison of the original scheme and enhanced schemes are shown in the figure below. The following figure also describes that the initialization content is the same as defined in V1.3, and the only modification is that it is transmitted using the SMC channel when a fast startup is required. "Fast Initialization Phase" below means that the sending of first sensor data is fast, but the sending of sensor initialization data will actually be much slower.

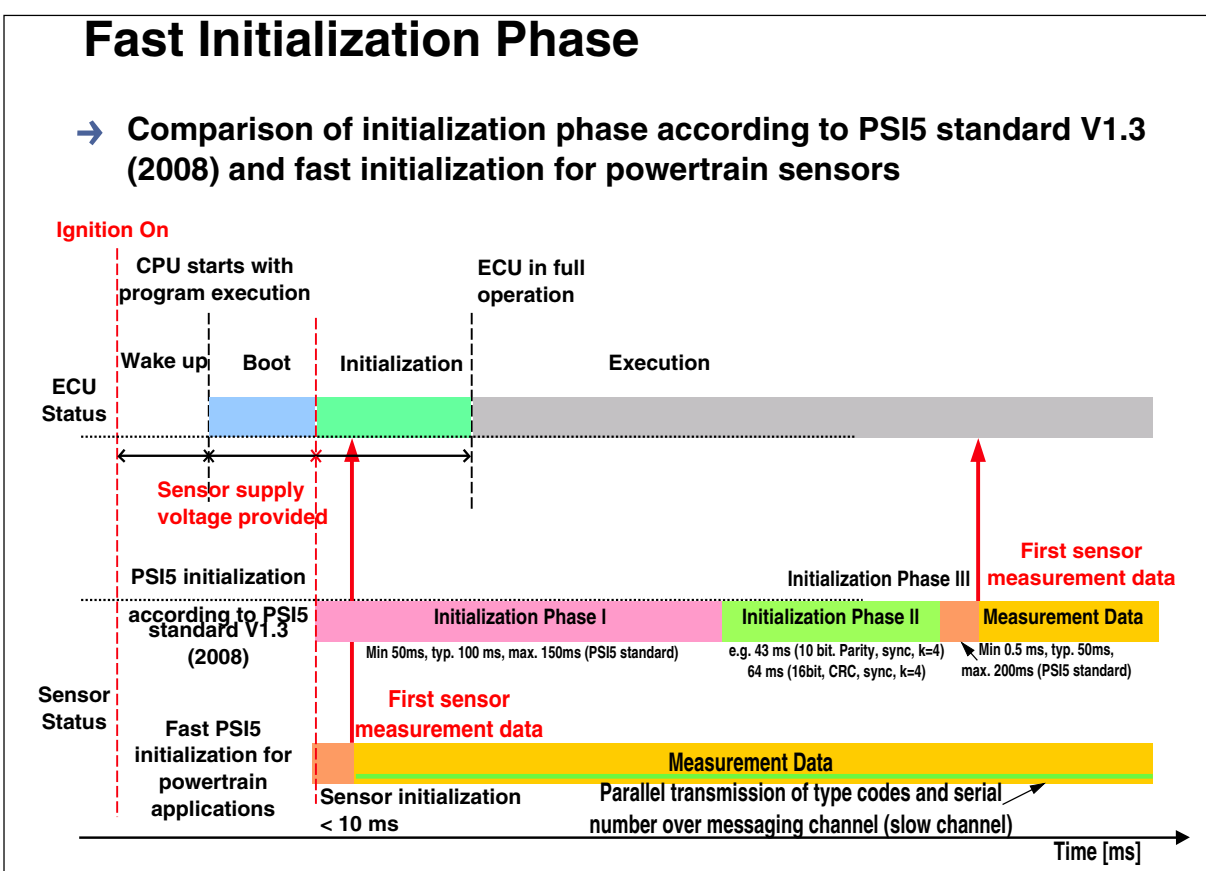


Figure 47-149. Fast initialization phase

47.7 Design Assumptions and Limitations

47.7.1 Assumptions

- Slot Configuration Register would always be programmed with valid number of bits for frame configuration.
- There will be no Timing error bits (T) in async mode. Only CRC Error (C) and Electrical Error (E) bits would be appended to message. There is no time out condition for the Manchester decoder in case of Async mode. 'NFR' bit is not set in Async mode.

47.7.2 Limitations

As this is a PSI5 receiver, designed primarily to only receive the PSI5/SMC data and store as it is into the RAM buffer. So, it would not be responsible neither to recognize the content of the received messages nor do any decision making based on the content. Similarly, for Command it would not recognize the content of command and would transmit as it is writing into corresponding DSR and reception of PSI5 messages would not in any way depend on the content of command as far as PSI receiver is concerned. Its the responsibility of SW to sequence the issue of commands across the PSI5 channels and thereby recognize the content of received message and infer it.

47.8 Miscellaneous descriptions

47.8.1 Internal SYNC Pulse generation and coordination across PSI5 channels

Each PSI5 channel has the Channel Target Pulse Registers (CTPR) and Channel Initialize Pulse Register (CIPR). The CTPR defines the period of the internal sync pulse generation and the CIPR defines the initial offset time with respect to the sync pulse trigger. The sync pulse trigger can be generated from either the PSI5_GCR[CTC_GED] global bit or the PSI5_PCCR[CTC_ED] channel specific bit.

When staggering of the channels is required then the PSI5_GCR[CTC_GED] global bit is used as a common trigger for all the PSI5 channels in which PSI5_PCCR[CTC_GED_SEL] bit is set to 1. Setting this bit to 1 triggers the internal sync pulse generators of all the PSI5 channels and the trigger of each channel is then automatically staggered with respect to the other channel, depending on the value of the CIPR.

When the specific PSI5 channel is not to be kept as a part of staggered channels, then the channel-specific PSI5_PCCR[CTC_GED_SEL] bit has to be kept as 0. The channel in this case, has to be triggered locally by writing a 1 to the PSI5_PCCR[CTC_ED] of that specific channel. This triggers the internal Sync Pulse Generator of that specific channel without affecting other channels.

Following is the procedure for starting the internal Sync Pulse Generators:

- If in Normal mode, then stop and reset all Channel Trigger Counters (CTCs) by setting the global PSI5_GCR[CTC_GED] bit to 0 or the channel specific PSI5_PCCR[CTC_ED] to 0.
- Enter Config mode.
- In Config mode, load all CTCs with the initialization value (offset) from CIPR.
- In Config mode, load all Counter Target Pulse Registers (CTPR) with the needed sync pulse period.
- Enter Normal mode.
- Send global start trigger (using PSI5_GCR[CTC_GED] bit) to all CTCs for synchronized start for the staggered channels OR send a local start trigger (by using the PSI5_PCCR[CTC_ED]) for channels not staggered.
- If the Compare Unit detects that the value in the CTC is equal to the value in CTPR, the CTC is automatically reset by hardware and restarts with 0.
- In Normal mode, if required, the CTC can be stopped and reset by writing a "0" to the PSI5_GCR[CTC_GED] or the PSI5_PCCR[CTC_ED] as the case may be.
- A sync pulse trigger is created. (See below.)

For the details about the operation of the below interface, please see Section 54.3.2.34, Counter Target Pulse Register (PSI5_CTPR), Section 54.3.2.35, Counter Initialize Pulse Register (PSI5_CIPR) and [Sync Pulse Trigger Generation](#).

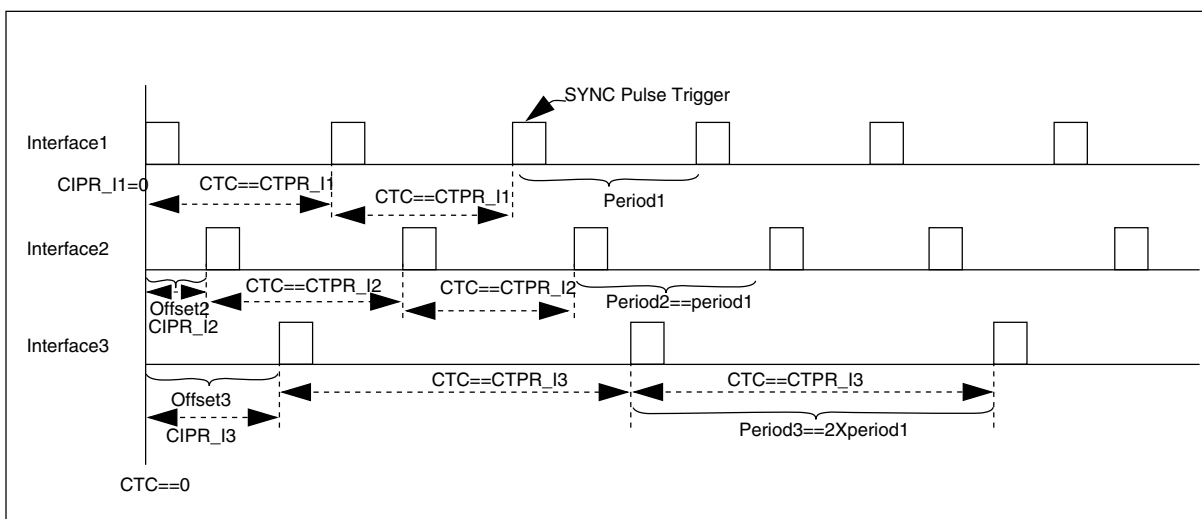


Figure 47-150. CIPR and CTPR

Figure 47-151 shows the various parameters (CIPR/CTPR, and so on) and the associated jitter, across different channels when these channels are staggered. (The figure also shows some other parameters, such as the modulated pulse width parameters and the Manchester decoder disable parameters.)

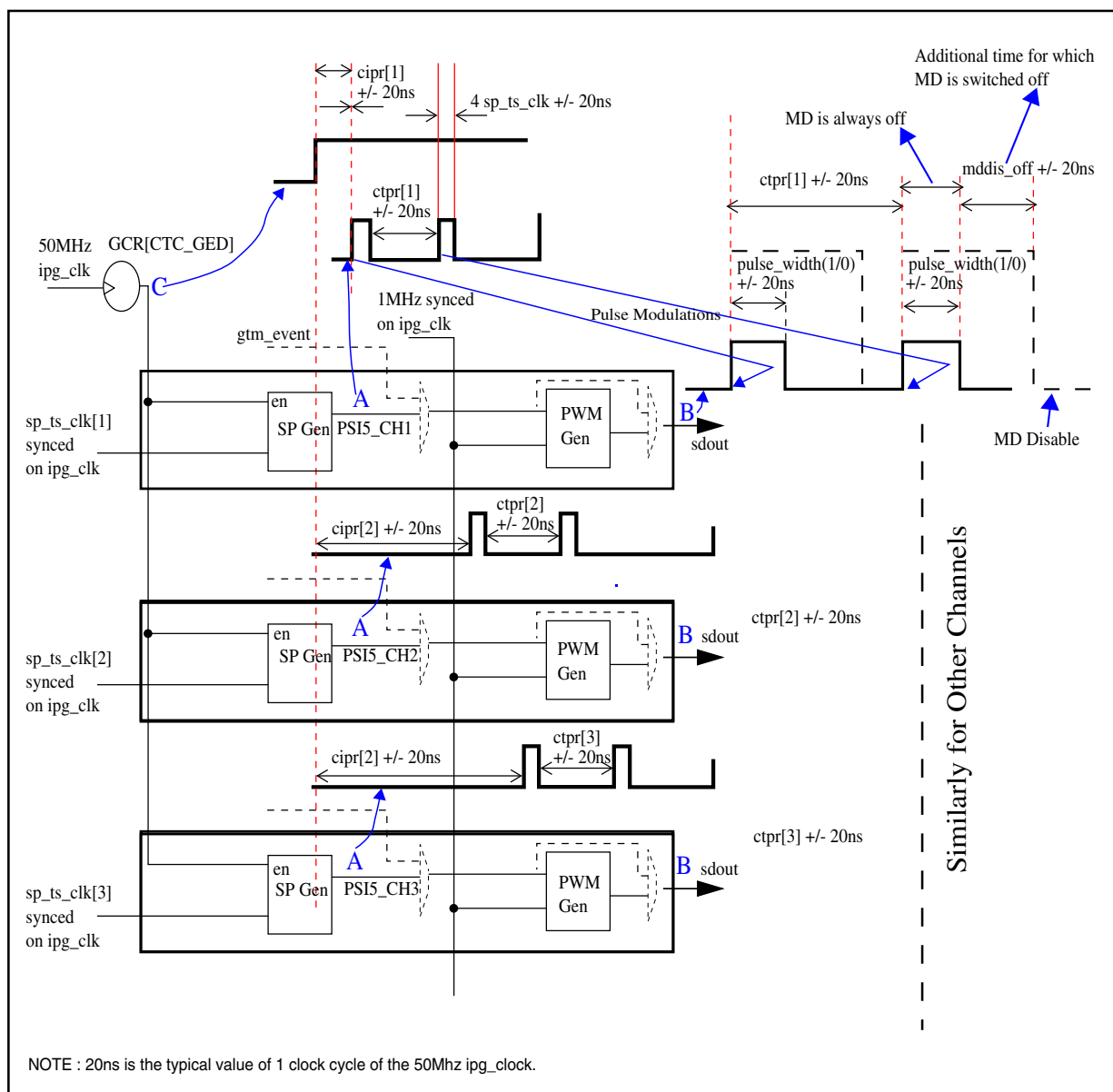
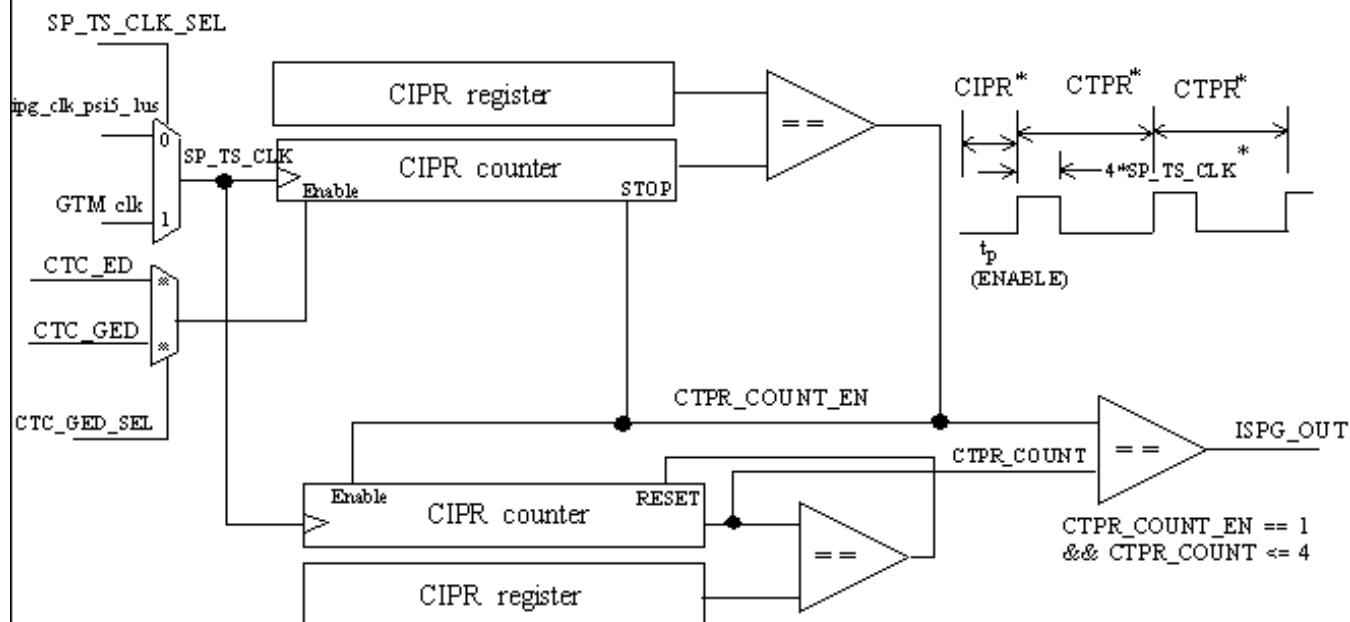


Figure 47-151. Sync Pulse Generation Parameters and Jitters

Figure 47-151 shows the various jitters.



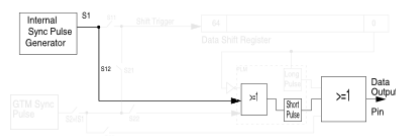
Note 1: All channels with CTC_GD selected will be enabled at the same time, enabling staggering of the ispg_out signals.

Note 2: This structure is replicated in each channel, with CTC_GED being a common signal to all channels.

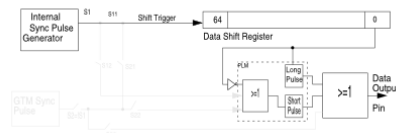
Figure 47-152. ISPG (Internal Sync Pulse Generator) Structure

47.8.2 Valid states for integrated sync pulse generator

State1: Periodic Sync Pulse Generation (without pulse length modulation)



State2: Periodic Sync Pulse Generation with ECU-to-sensor Communication



State3: Event triggered (e.g. angle synchronous) sync pulses



Figure 47-153. ECU to Sensor Communication States

State1: Periodic Sync Pulse Generation (without pulse length modulation)

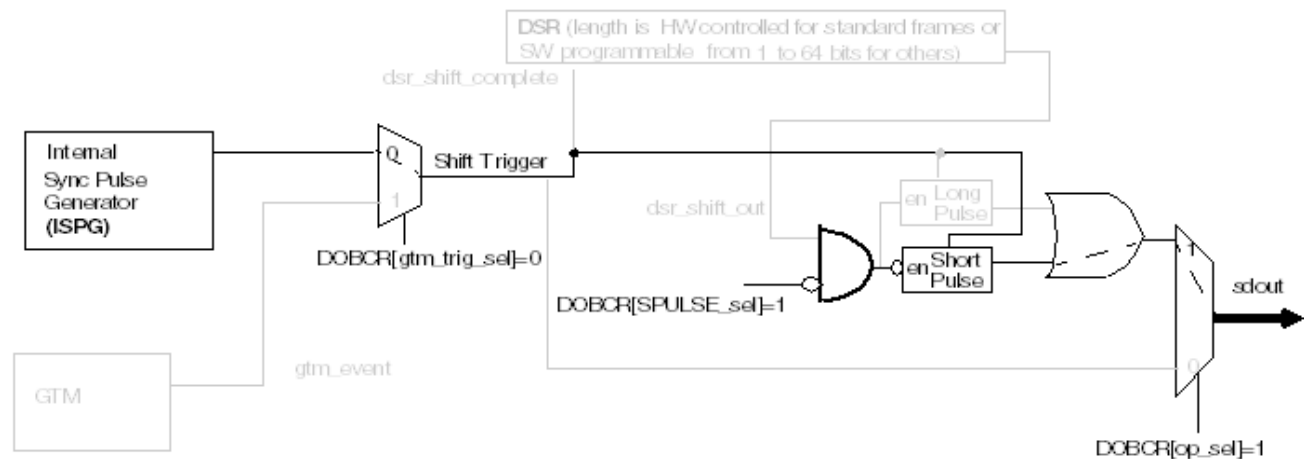


Figure 47-154. ECU to Sensor Communication States - State 1

State2: Periodic Sync Pulse Generation with ECU-to-sensor Communication

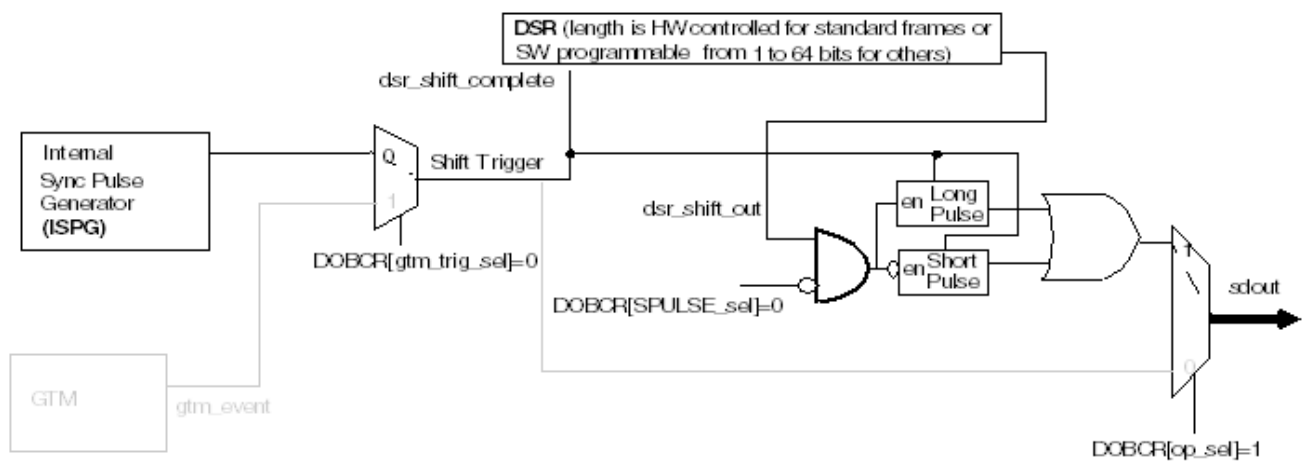


Figure 47-155. ECU to Sensor Communication States - State 2

State3: Event triggered (e.g. angle synchronous) sync pulses

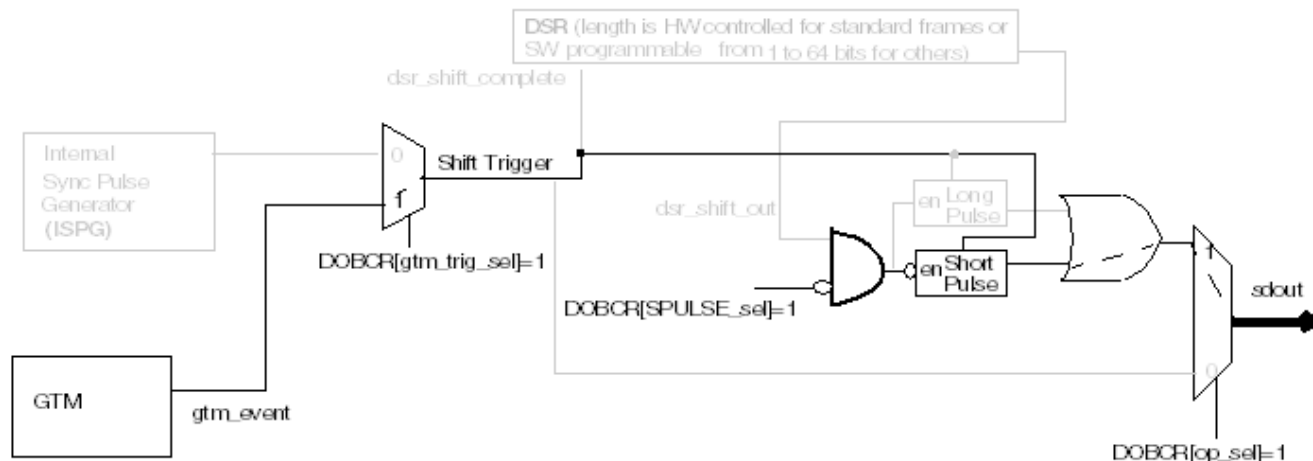


Figure 47-156. ECU to Sensor Communication States - State 3

State4: Event triggered sync pulse, including ECU-to-sensor communication

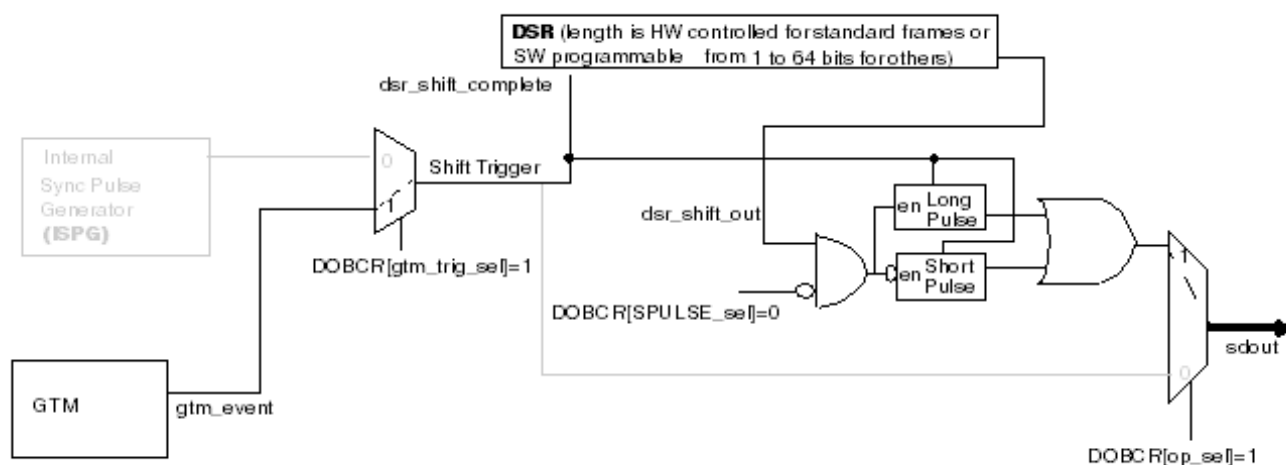


Figure 47-157. ECU to Sensor Communication States - State 4

State5: Event triggered sync pulses, direct access

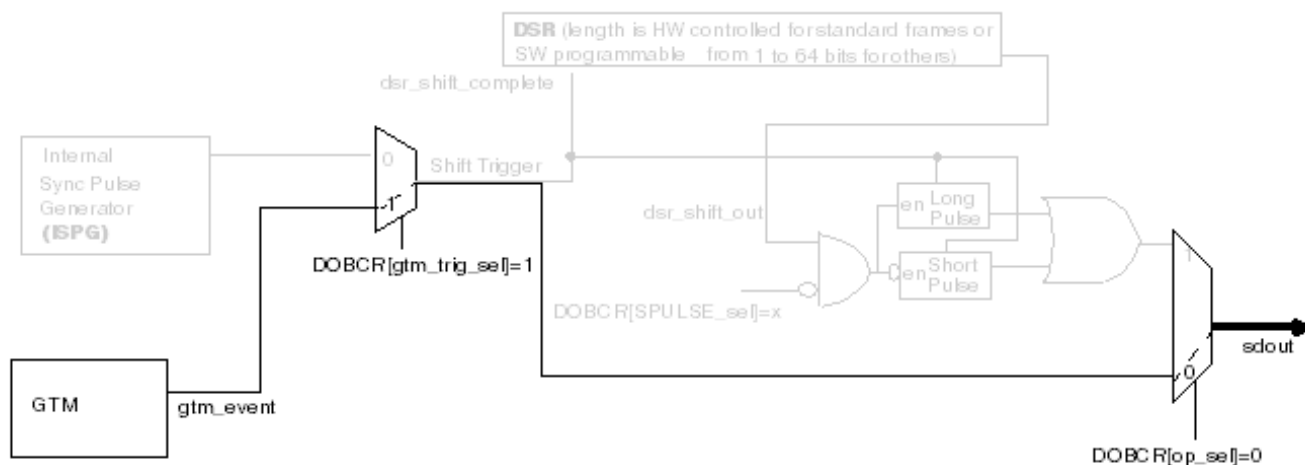


Figure 47-158. ECU to Sensor Communication States - State 5



Chapter 48

SENT Receiver (SRX)

48.1 Chip-specific SRX information

This chip has two instances of the SRX module.

Table 48-1. SRX module instances

Instance	Input/receive channels	FIFO depth
SRX_0	6	16
SRX_1	6	16

48.2 Introduction

The Single Edge Nibble Transmission (SENT) Receiver (SRX) module is a multi-channel receiver for receiving serial data frames which are being transmitted by a sensor implementing the SENT encoding scheme and present them to the CPU for further processing.

This module is based on the SAE J2716 information report titled "SENT - Single Edge Nibble Transmission for Automotive Applications" and released on January 27, 2010 (<http://www.sae.org>), Apr2007 and Feb2008. As per this standard, the SENT protocol is intended for use in applications where high resolution sensor data needs to be communicated from a sensor to an Engine Control Unit (ECU). It is intended as a replacement for the lower resolution methods of 10 bit A/Ds and PWM and as a simpler low-cost alternative to CAN or LIN. The implementation assumes that the sensor is a smart sensor containing a microprocessor or dedicated logic device (ASIC) to create the signal.

The SENT encoding scheme is a unidirectional communications scheme from the sensor / transmitting device to the controller /receiving device which does not include a coordination signal from the controller/receiving device. The sensor signal is transmitted as a series of pulses with data encoded as falling to falling edge periods.

48.2.1 Features

The SENT receiver module has the following features:

- The number of SENT channels supported are device-specific. See the device configuration details.
- Unified message read logic to transfer all messages received to system memory via interrupts or DMA
 - Programmable option per channel to support reading of messages via DMA or Interrupt
 - Internal round robin arbitration sequencer to loop across channels to read messages from each channel leaving CPU/DMA transparent to this process
 - A FIFO for storing the received Fast Messages from all channels that are to be read out via DMA. FIFO depth is device-specific. See the device configuration section for details.
 - All received messages are stored excluding the synchronization pulse (for Fast messages) and identifiers (for Slow Serial Messages)
 - Separate buffers for storing Fast and Slow Serial Messages
 - Separate DMA and Interrupts generated for Fast and Slow Serial Messages
- Supports compensation for variation in SENT Tx Clock up to $\pm 25\%$ and adjusts its measurement for drift in the Transmitter and Receiver clocks per channel
 - Calibration pulse correction factor per channel is available for use by software
- Implements for each channel all receiver diagnostics as specified by the SAE Specifications
- Internal 4-bit CRC calculator that can be configured to compute CRC in both Legacy and Recommended method (as given in SAE SENT Specification) for both Fast and Short Serial Message per channel
- Internal 6-bit CRC calculator for Enhanced Serial Messages per channel

- Supports all bit rates by having per channel configurable Rx clock periods from 3 μ s to 90 μ s
- Support for configurable number of data nibbles per channel
- Option to disable individual channel via programmable control bit
- Time stamping on start of each valid message (i.e. without errors) across all channels. This time stamp is appended to each message for synchronization of messages and for application to understand how much time has elapsed between successive sensor readings
- Support for pause pulse per channel. This can be enabled by programming a bit in the control register for that channel
- Input Programmable Filter on each channel to filter any glitches on input. This filter duration is programmable by the user software.
- 32-bit register interface for programming the module's registers and reading the received messages
- Works on two clocks, one for accurate receive operations (high frequency receiver clock or protocol clock) and other for message read logic (system bus clock) and the module can work irrespective of the frequency relationship between the two clocks. See [Clocks and resets](#) for more details on the clocks.

48.2.2 Modes of operation

Modes of operation.

48.2.2.1 Debug mode

In debug mode, message reception will be halted (without waiting for the current message being received to be completed) and all existing status and messages will be retained. DBG_FRZ bit is required to be set for this mode. On debug exit, a calibration length error (CAL_LEN_ERR) can be reported and the channels will start receiving messages from a valid calibration pulse.

48.2.2.2 Low power modes

SENT Receiver behavior is affected by chip-specific low power modes (for example, STOP or WAIT modes). These low power modes should either gate-off both the system bus clock and the protocol clocks or reduce their frequencies, depending on the application.

On exiting from these low power modes, calibration length error (CAL_LEN_ERR) can get set in channel status registers as the SENT sensor (transmitter) could be in the middle of a message transmission at that point. Thus, software should clear all error status bits after exiting the low power modes.

Modes where both system bus clock and protocol clock are gated: In this case, both message reception and interrupt / DMA request generation will halt. The internal receiver channels will freeze (since clock is not there) and will not receive any messages. Messages which are completely received before entering the low power mode can be read after the low power mode exit.

Modes where frequency of system bus clock is changed: For reduced power consumption the system bus clock can be reduced but not below a minimum value (See [System bus clock requirements](#)) to avoid overflow of received messages. Protocol clock should not go below the minimum value (See [High frequency receiver clock \(protocol clock\) requirements](#)) for accurate receive operations.

Note

Gating of clocks in the low power mode is chip-specific, see the chapter that refers how the modules behave in low power modes.

NOTE

If the SOC requests entry to debug or stop mode, the message being received currently is discarded and the SOC enters debug/stop mode immediately. This does not affect the messages received completely prior to entering debug mode and will still be present on the message buffer and registers until they are read out.

NOTE

The message read registers [Channel 'n' Fast Message Data Read Register (n = 0 to (CH-1)) (CHn_FMSG_DATA), Channel 'n' Fast Message CRC Read Register (n = 0 to (CH-1)) (CHn_FMSG_CRC), Channel 'n' Fast Message Time Stamp Read Register (n = 0 to (CH-1)) (CHn_FMSG_TS), Channel 'n' Serial Message Read Register (Bit 3) (n = 0 to (CH-1))

(CHn_SMSG_BIT3), Channel 'n' Serial Message Read Register (Bit 2) (n = 0 to (CH-1)) (CHn_SMSG_BIT2), Channel 'n' Serial Message Time Stamp Read Register (n = 0 to (CH-1)) (CHn_SMSG_TS), DMA Fast Message Data Read Register (DMA_FMSG_DATA), DMA Fast Message CRC Read Register (DMA_FMSG_CRC), DMA Fast Message Time Stamp Read Register (DMA_FMSG_TS), DMA Slow Serial Message Bit3 Read Register (DMA_SMSG_BIT3), DMA Slow Serial Message Bit2 Read Register (DMA_SMSG_BIT2) and DMA Slow Serial Message Time Stamp Read Register (DMA_SMSG_TS)] will appear to lose their contents in the following conditions:

- The very first message is being received but not yet completely received and the MCU enters debug or freeze mode, the current message reception will get discarded and message read registers (as mentioned above) will read zeros
- Auto clear functionality is enabled (GBL_CTRL[FAST_CLR] = 1). In this case, when first message is read, it will get clear the message read registers (due to auto clear functionality being enabled). On reading again, the message read registers (as mentioned above) might read zeros.

If the MCU requests entry to debug or stop mode, the message being received currently by SENT Receiver is discarded and the MCU enters debug/stop mode immediately. This does not affect the messages received completely, prior to entering debug mode and these messages will still be present on the message buffer and registers until they are read out. Thus allow one message to be received completely and do not enable "Auto Clear" (GBL_CTRL[FAST_CLR] = 0), to allow messages to be read in debug or stop mode.

48.2.3 Block diagram

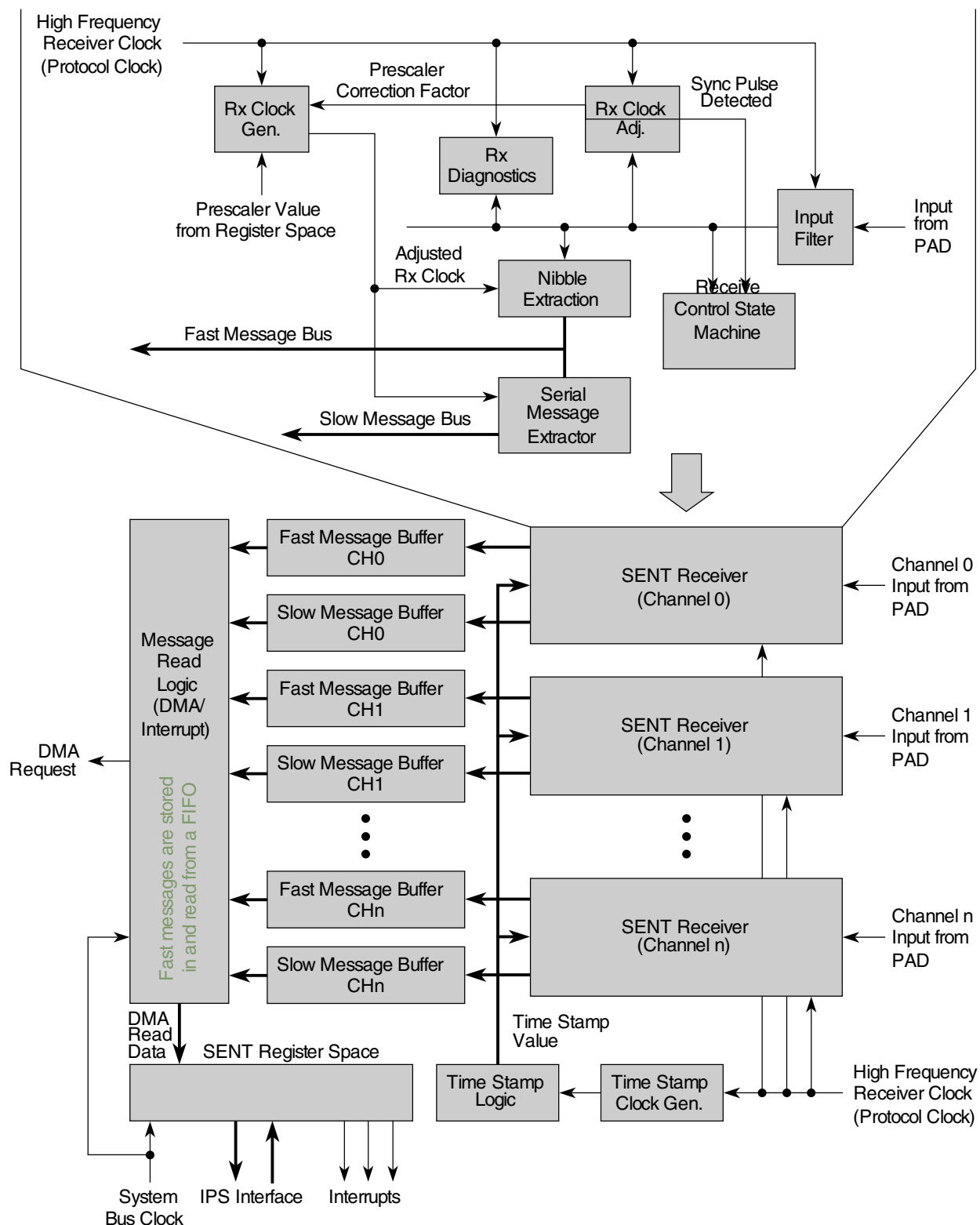


Figure 48-1. SENT Receiver block diagram

48.2.4 Design overview

The SENT Receiver Module is a unified receiver module comprising of a number of receiver channels and a unified DMA and Register Space for accessing the messages from each channel. [Figure 48-1](#) shows the top level block diagram for the SENT Receiver module.

Each channel of the SENT Receiver is composed of a Clock Generation and Adjustment block to generate the receiver clock and adjust it for frequency variation in the transmitter clock. This block also detects the synchronization/calibration pulse which is present in every SENT message.

The SENT Receiver State Machine controls the sequencing of decoding the messages and combining the information as nibbles. In parallel a diagnostic block is running that monitors the incoming messages for any errors and also verifies the CRC of the received messages. The fast and serial message extractors collate the message nibbles and add the channel number and time stamp information to the messages. A free running time stamp counter is used to add the time stamp value to each message received. The value of this counter is input to each channel and the channel logic appends the time stamp to each message. These messages are then formatted to be stored in the DMA FIFOs (for channels which are selected to be connected to the DMA) or in corresponding message read buffer. Only the messages that pass the diagnostics checks are forwarded to be stored in the FIFOs or buffers.

The SENT Receiver provides two mechanisms for reading the received messages, which are via DMA transfers and via Interrupts. Based on the control bit settings in the control registers, a DMA request or Interrupt is asserted to signal the DMA controller or CPU that messages are available to be read.

The free running time stamp counter, register block and DMA interface are common for all channels configured. The register space contains all the necessary registers for all channels. The register space is 32-bit aligned and accessible via 8-/16-/32-bit accesses. The registers for all un-configured channels are marked as reserved.

48.2.4.1 Functional overview

After reset, the SENT module comes up in the disabled state. All channel enable bits and the global enable bit are set to 0 and the user software must program the SENT module parameters correctly before enabling it. The user software must correctly configure all parameters to avoid any underflow or overflow in the message FIFOs or buffers.

Once enabled, the Receiver Clock Generation and Adjustment block detects the synchronization pulse. Until then, the state machine blocks all subsequent operations of the receiver. Once the synchronization pulse is detected, the receiver clock tick period is compensated for variation in transmitter clock tick period and the state machine then waits for the reception of data nibbles. The adjusted receiver clock is used to sample the data nibbles from the incoming serial message. The state machine waits until the status and communication nibble and configured number of data nibbles are received. The state machine then waits for the pause pulse (if configured to receive a pause pulse). The fast and serial message extractor block provides the state machine with the information on how many nibbles have been received. All diagnostics checks are computed in parallel. Once all required data nibbles are received, the CRC is checked, time stamp and channel number is added to the message. The time stamp for a Fast Message is sampled at the falling edge that comes at the end of the calibration pulse (which is also the start of the status and communication nibble pulse) and is stored in the message read buffer along with the message. This final message is stored in the DMA FIFO or message buffer as described in the subsequent sections. Any error detected during the receive process causes the status bits in the corresponding channel's status register to be set.

The above process repeats across each enabled channel in parallel and keeps pushing error free messages into the FIFOs or buffers on high frequency receiver clock. The module's message read logic synchronizes all these messages on the bus clock and pops out these messages on any DMA or CPU read access. Any DMA underflow errors are stored in the global status register while the overflow errors are stored in the corresponding channel's status register. In case of an overflow in Fast Messages, the newer messages received will overwrite the previous message.

For detailed description on the complete functionality of each block, see [Functional description](#).

48.3 External signal description

This section lists all inputs to the SENT Receiver block.

48.3.1 SENT_RX [(CH-1):0]

These pins serve as the serial data input for each of the SENT Receiver channel. There is one bit per channel supported by the device. The SENT sensor input pads on the device using this module should have pull-up enabled to ensure a 1'b1 is seen at the module input when SENT sensor is not driving any data.

Note

The value of CH is device-specific. See the device configuration details.

48.4 Memory map and register definition

This section provides a detailed description of all registers accessible in the SENT Receiver block.

The following table gives an overview on all SENT Receiver registers. The memory map comprises of 32-bit aligned registers which can be accessed via 8-bit, 16-bit, or 32-bit accesses. Write access to reserved locations will generate transfer error. Read access to reserved locations will also generate transfer error and the read data bus will show all 0s. The Memory Map and complete module is in Big Endian Format.

The module will not check for correctness of programmed values in Register Space and software must ensure that correct values are being written.

NOTE

Register Address = Base Address + Address Offset, where the Base Address is defined at the MCU level and the Address Offset is defined at the module level. The number of registers are device specific. See the device configuration section for details.

NOTE

The value of the CH is device-specific. See the device configuration section for details.

SRX memory map

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
0	Global Control Register (SRX_GBL_CTRL)	32	R/W	0001_0010h	48.4.1/2623
4	Channel Enable Register (SRX_CHNL_EN)	32	R/W	0000_0000h	48.4.2/2625
8	Global Status Register (SRX_GBL_STATUS)	32	w1c	0000_0000h	48.4.3/2626
C	Fast Message Ready Status Register (SRX_FMSG_RDY)	32	R/W	0000_0000h	48.4.4/2627
10	Slow Serial Message Ready Status Register (SRX_SMSG_RDY)	32	R/W	0000_0000h	48.4.5/2628
18	Data Control Register 1 (SRX_DATA_CTRL1)	32	R/W	1111_1100h	48.4.6/2629

Table continues on the next page...

SRX memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
28	Fast Message DMA Control Register (SRX_FDMA_CTRL)	32	R/W	0000_0000h	48.4.7/2631
2C	Slow Serial Message DMA Control Register (SRX_SDMA_CTRL)	32	R/W	0000_0000h	48.4.8/2631
34	Fast Message Ready Interrupt Control Register (SRX_FRDY_IE)	32	R/W	0000_0000h	48.4.9/2632
38	Slow Serial Message Ready Interrupt Enable Register (SRX_SRDY_IE)	32	R/W	0000_0000h	48.4.10/2633
40	DMA Fast Message Data Read Register (SRX_DMA_FMSG_DATA)	32	R	0000_0000h	48.4.11/2633
44	DMA Fast Message CRC Read Register (SRX_DMA_FMSG_CRC)	32	R	0000_0000h	48.4.12/2634
48	DMA Fast Message Time Stamp Read Register (SRX_DMA_FMSG_TS)	32	R	0000_0000h	48.4.13/2635
50	DMA Slow Serial Message Bit3 Read Register (SRX_DMA_SMSG_BIT3)	32	R	0000_0000h	48.4.14/2635
54	DMA Slow Serial Message Bit2 Read Register (SRX_DMA_SMSG_BIT2)	32	R	0000_0000h	48.4.15/2637
58	DMA Slow Serial Message Time Stamp Read Register (SRX_DMA_SMSG_TS)	32	R	0000_0000h	48.4.16/2637
60	Channel 'n' Clock Control Register (n = 0 to (CH-1)) (SRX_CH0_CLK_CTRL)	32	R/W	0000_8000h	48.4.17/2638
64	Channel 'n' Status Register (n=0 to (CH-1)) (SRX_CH0_STATUS)	32	w1c	0000_0000h	48.4.18/2640
68	Channel 'n' Configuration Register (n=0 to (CH-1)) (SRX_CH0_CONFIG)	32	R/W	0000_0104h	48.4.19/2643
70	Channel 'n' Clock Control Register (n = 0 to (CH-1)) (SRX_CH1_CLK_CTRL)	32	R/W	0000_8000h	48.4.17/2638
74	Channel 'n' Status Register (n=0 to (CH-1)) (SRX_CH1_STATUS)	32	w1c	0000_0000h	48.4.18/2640
78	Channel 'n' Configuration Register (n=0 to (CH-1)) (SRX_CH1_CONFIG)	32	R/W	0000_0104h	48.4.19/2643
80	Channel 'n' Clock Control Register (n = 0 to (CH-1)) (SRX_CH2_CLK_CTRL)	32	R/W	0000_8000h	48.4.17/2638
84	Channel 'n' Status Register (n=0 to (CH-1)) (SRX_CH2_STATUS)	32	w1c	0000_0000h	48.4.18/2640
88	Channel 'n' Configuration Register (n=0 to (CH-1)) (SRX_CH2_CONFIG)	32	R/W	0000_0104h	48.4.19/2643
90	Channel 'n' Clock Control Register (n = 0 to (CH-1)) (SRX_CH3_CLK_CTRL)	32	R/W	0000_8000h	48.4.17/2638
94	Channel 'n' Status Register (n=0 to (CH-1)) (SRX_CH3_STATUS)	32	w1c	0000_0000h	48.4.18/2640
98	Channel 'n' Configuration Register (n=0 to (CH-1)) (SRX_CH3_CONFIG)	32	R/W	0000_0104h	48.4.19/2643
A0	Channel 'n' Clock Control Register (n = 0 to (CH-1)) (SRX_CH4_CLK_CTRL)	32	R/W	0000_8000h	48.4.17/2638

Table continues on the next page...

SRX memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
A4	Channel 'n' Status Register (n=0 to (CH-1)) (SRX_CH4_STATUS)	32	w1c	0000_0000h	48.4.18/2640
A8	Channel 'n' Configuration Register (n=0 to (CH-1)) (SRX_CH4_CONFIG)	32	R/W	0000_0104h	48.4.19/2643
B0	Channel 'n' Clock Control Register (n = 0 to (CH-1)) (SRX_CH5_CLK_CTRL)	32	R/W	0000_8000h	48.4.17/2638
B4	Channel 'n' Status Register (n=0 to (CH-1)) (SRX_CH5_STATUS)	32	w1c	0000_0000h	48.4.18/2640
B8	Channel 'n' Configuration Register (n=0 to (CH-1)) (SRX_CH5_CONFIG)	32	R/W	0000_0104h	48.4.19/2643
160	Channel 'n' Fast Message Data Read Register (n=0 to (CH-1)) (SRX_CH0_FMSG_DATA)	32	R	0000_0000h	48.4.20/2646
164	Channel 'n' Fast Message CRC Read Register (n=0 to (CH-1)) (SRX_CH0_FMSG_CRC)	32	R	0000_0000h	48.4.21/2647
168	Channel 'n' Fast Message Time Stamp Read Register (n=0 to (CH-1)) (SRX_CH0_FMSG_TS)	32	R	0000_0000h	48.4.22/2647
16C	Channel 'n' Serial Message Read Register (Bit 3) (n=0 to (CH-1)) (SRX_CH0_SMSG_BIT3)	32	R	0000_0000h	48.4.23/2648
170	Channel 'n' Serial Message Read Register (Bit 2)(n=0 to (CH-1)) (SRX_CH0_SMSG_BIT2)	32	R	0000_0000h	48.4.24/2649
174	Channel 'n' Serial Message Time Stamp Read Register (n=0 to (CH-1)) (SRX_CH0_SMSG_TS)	32	R	0000_0000h	48.4.25/2650
178	Channel 'n' Fast Message Data Read Register (n=0 to (CH-1)) (SRX_CH1_FMSG_DATA)	32	R	0000_0000h	48.4.20/2646
17C	Channel 'n' Fast Message CRC Read Register (n=0 to (CH-1)) (SRX_CH1_FMSG_CRC)	32	R	0000_0000h	48.4.21/2647
180	Channel 'n' Fast Message Time Stamp Read Register (n=0 to (CH-1)) (SRX_CH1_FMSG_TS)	32	R	0000_0000h	48.4.22/2647
184	Channel 'n' Serial Message Read Register (Bit 3) (n=0 to (CH-1)) (SRX_CH1_SMSG_BIT3)	32	R	0000_0000h	48.4.23/2648
188	Channel 'n' Serial Message Read Register (Bit 2)(n=0 to (CH-1)) (SRX_CH1_SMSG_BIT2)	32	R	0000_0000h	48.4.24/2649
18C	Channel 'n' Serial Message Time Stamp Read Register (n=0 to (CH-1)) (SRX_CH1_SMSG_TS)	32	R	0000_0000h	48.4.25/2650
190	Channel 'n' Fast Message Data Read Register (n=0 to (CH-1)) (SRX_CH2_FMSG_DATA)	32	R	0000_0000h	48.4.20/2646
194	Channel 'n' Fast Message CRC Read Register (n=0 to (CH-1)) (SRX_CH2_FMSG_CRC)	32	R	0000_0000h	48.4.21/2647
198	Channel 'n' Fast Message Time Stamp Read Register (n=0 to (CH-1)) (SRX_CH2_FMSG_TS)	32	R	0000_0000h	48.4.22/2647
19C	Channel 'n' Serial Message Read Register (Bit 3) (n=0 to (CH-1)) (SRX_CH2_SMSG_BIT3)	32	R	0000_0000h	48.4.23/2648
1A0	Channel 'n' Serial Message Read Register (Bit 2)(n=0 to (CH-1)) (SRX_CH2_SMSG_BIT2)	32	R	0000_0000h	48.4.24/2649

Table continues on the next page...

SRX memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
1A4	Channel 'n' Serial Message Time Stamp Read Register (n=0 to (CH-1)) (SRX_CH2_SMSG_TS)	32	R	0000_0000h	48.4.25/2650
1A8	Channel 'n' Fast Message Data Read Register (n=0 to (CH-1)) (SRX_CH3_FMSG_DATA)	32	R	0000_0000h	48.4.20/2646
1AC	Channel 'n' Fast Message CRC Read Register (n=0 to (CH-1)) (SRX_CH3_FMSG_CRC)	32	R	0000_0000h	48.4.21/2647
1B0	Channel 'n' Fast Message Time Stamp Read Register (n=0 to (CH-1)) (SRX_CH3_FMSG_TS)	32	R	0000_0000h	48.4.22/2647
1B4	Channel 'n' Serial Message Read Register (Bit 3) (n=0 to (CH-1)) (SRX_CH3_SMSG_BIT3)	32	R	0000_0000h	48.4.23/2648
1B8	Channel 'n' Serial Message Read Register (Bit 2)(n=0 to (CH-1)) (SRX_CH3_SMSG_BIT2)	32	R	0000_0000h	48.4.24/2649
1BC	Channel 'n' Serial Message Time Stamp Read Register (n=0 to (CH-1)) (SRX_CH3_SMSG_TS)	32	R	0000_0000h	48.4.25/2650
1C0	Channel 'n' Fast Message Data Read Register (n=0 to (CH-1)) (SRX_CH4_FMSG_DATA)	32	R	0000_0000h	48.4.20/2646
1C4	Channel 'n' Fast Message CRC Read Register (n=0 to (CH-1)) (SRX_CH4_FMSG_CRC)	32	R	0000_0000h	48.4.21/2647
1C8	Channel 'n' Fast Message Time Stamp Read Register (n=0 to (CH-1)) (SRX_CH4_FMSG_TS)	32	R	0000_0000h	48.4.22/2647
1CC	Channel 'n' Serial Message Read Register (Bit 3) (n=0 to (CH-1)) (SRX_CH4_SMSG_BIT3)	32	R	0000_0000h	48.4.23/2648
1D0	Channel 'n' Serial Message Read Register (Bit 2)(n=0 to (CH-1)) (SRX_CH4_SMSG_BIT2)	32	R	0000_0000h	48.4.24/2649
1D4	Channel 'n' Serial Message Time Stamp Read Register (n=0 to (CH-1)) (SRX_CH4_SMSG_TS)	32	R	0000_0000h	48.4.25/2650
1D8	Channel 'n' Fast Message Data Read Register (n=0 to (CH-1)) (SRX_CH5_FMSG_DATA)	32	R	0000_0000h	48.4.20/2646
1DC	Channel 'n' Fast Message CRC Read Register (n=0 to (CH-1)) (SRX_CH5_FMSG_CRC)	32	R	0000_0000h	48.4.21/2647
1E0	Channel 'n' Fast Message Time Stamp Read Register (n=0 to (CH-1)) (SRX_CH5_FMSG_TS)	32	R	0000_0000h	48.4.22/2647
1E4	Channel 'n' Serial Message Read Register (Bit 3) (n=0 to (CH-1)) (SRX_CH5_SMSG_BIT3)	32	R	0000_0000h	48.4.23/2648
1E8	Channel 'n' Serial Message Read Register (Bit 2)(n=0 to (CH-1)) (SRX_CH5_SMSG_BIT2)	32	R	0000_0000h	48.4.24/2649
1EC	Channel 'n' Serial Message Time Stamp Read Register (n=0 to (CH-1)) (SRX_CH5_SMSG_TS)	32	R	0000_0000h	48.4.25/2650

48.4.1 Global Control Register (SRX_GBL_CTRL)

This register provides the global control and setting for the SENT Receiver Module. The SENT_EN bit must be programmed after all other modules settings and control bits have been written to.

Address: 0h base + 0h offset = 0h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	TSPRSC							0	FIFOWM							
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1
Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	NIB_LEN_VAR_LIMIT	0				FMFOIE	FMDUIE	SMDUIE	0	FIFO_EN	0	FAST_CLR	0	DBG_FRZ	0	SENT_EN
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0

SRX_GBL_CTRL field descriptions

Field	Description
0–7 TSPRSC	Time Stamp Prescaler Value. Value in number of clock cycles of high frequency receiver clock set in order to obtain a clock frequency for the time stamp counter block. 0 to 255 - Time Stamp Clock = (High Frequency Clock)/(Value + 1). Value set by user software should be such that the clock period is at least 1 μ s. If the high frequency clock input is 75 MHz, then the value set in this field would be 0x4A (or 74) to obtain a 1 μ s clock. Note: Since the time stamp clock is common for all channels, the timestamp clock frequency must be at least twice the fastest receiver channel in the application. This is to ensure that the time stamp clock is fast enough to show change in timestamp from message to message.
8–10 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
11–15 FIFOWM	DMA Read FIFO Water Mark Level. Indicates the number of Fast Messages to be stored in FIFO before a DMA request is asserted. Default = 1 message. 0 Invalid Value. Should not be used 1 to 31 Valid water mark levels. Value fixed by user software. NOTE: The number of bytes to be transferred set in the DMA Controller's TCD must be set equal to the number of bytes corresponding to the set water mark value.

Table continues on the next page...

SRX_GBL_CTRL field descriptions (continued)

Field	Description
16 NIB_LEN_VAR_LIMIT	Decides the allowable jitter/clock drift limit for nibble length measurement. NOTE: This bit should be set to 1 for proper functioning of Sent Rx. 0 Allows a jitter/clock drift of approx upto 12.5% of transmitter clock tick period. This actual allowed jitter/drift value is dependent on the transmitter clock tick period and HF receiver clock period. 1 Allows a jitter/clock drift as per SAE spec (50%).
17–20 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
21 FMFOIE	Fast Message FIFO Overflow Interrupt Enable: Enables interrupt assertion when Fast Message is received on a channel which has DMA enabled after FIFO is full. Default value is 0. 0 Interrupt is disabled 1 Interrupt is enabled
22 FMDUIE	Fast Message DMA Underflow Interrupt Enable: Enables interrupt assertion when there is an underflow condition on Fast Message DMA registers. Default value is 0. 0 Interrupt is disabled 1 Interrupt is enabled
23 SMDUIE	Slow Serial Message DMA Underflow Interrupt Enable: Enables interrupt assertion when there is an underflow condition on Slow Message DMA registers. Default value is 0. 0 Interrupt is disabled 1 Interrupt is enabled
24 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
25 FIFO_EN	FIFO enable bit. Enables FIFO when DMA is enabled. When this bit is low, data is read from module as explained in DMA request for Fast Message reading when FIFO is disabled and when the bit is high, data is read from module as explained in DMA request for Fast Message reading via FIFO . Default value is 0. 0 FIFO is disabled 1 FIFO is enabled
26 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
27 FAST_CLR	Fast Clearing Enable bit. Enables clearing of ready status (for both fast and slow) bit automatically when corresponding message is read. Default value is 1. See Fast Message Ready Status Register (SRX_FMSG_RDY) and Slow Serial Message Ready Status Register (SRX_SMSG_RDY) for details on the ready status bits. 0 Fast Clearing is disabled. Bits will be cleared by writing 1 1 Fast Clearing is enabled
28 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
29 DBG_FRZ	Debug Freeze. This bit will enable the debug mode support. SENT Module will freeze in debug mode if this bit is set. Default is 0. 0 No effect in debug mode 1 Freeze in debug mode

Table continues on the next page...

SRX_GBL_CTRL field descriptions (continued)

Field	Description
30 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
31 SENT_EN	SENT Receiver Global Enable. This bit enables or disables the complete receiver module irrespective of the setting of individual channel enable bits. User software must program all channel parameters before setting this bit. Default is 0. 0 Entire module is disabled 1 Module is enabled

48.4.2 Channel Enable Register (SRX_CHNL_EN)

This register is used to enable individual SENT Receiver channels in the module. The user software must ensure all channel parameters and control settings have been programmed before writing to this bit. In case these parameters are changed while the channel is enabled, normal operation is not guaranteed.

NOTE

Reads of bits beyond those for the supported number of channels should be ignored, and these bits must be written to 0.

Address: 0h base + 4h offset = 4h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0																									EN_CH						
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

SRX_CHNL_EN field descriptions

Field	Description
0–25 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
26–31 EN_CH	Enable bits for channels 5 to 0 . All channels are disabled after reset. User software must program all channel parameters before setting this bit. Once set, other channel parameters in registers should not be changed until this bit is cleared. Only Interrupt Enable and Status bits can be written to. 0 Channel is disabled 1 Channel is enabled. Channel parameter registers are locked for writing

48.4.3 Global Status Register (SRX_GBL_STATUS)

This register provides the status of various conditions in the module that are common across all channels.

Address: 0h base + 8h offset = 8h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0					FMFO	FMDU	SMDU	0							
W						w1c	w1c	w1c								
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

SRX_GBL_STATUS field descriptions

Field	Description
0–20 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
21 FMFO	Fast Message FIFO Overflow: Interrupt status bit which indicates that Fast Message is received on any channel which has DMA enabled after FIFO is full. Default value is 0. This bit is valid only when fifo is enabled 0 No Overflow 1 Overflow occurred
22 FMDU	Fast Message DMA Underflow: Interrupt Status bit that indicates when there is an underflow in Fast Message buffers or FIFO in any channel which has DMA enabled. If FIFO is enabled, it is set when FIFO is empty and software still reads Fast Message DMA registers. And if FIFO is not enabled, it is set when there is no message in the DMA buffers for read and the software still reads the Fast Message DMA register. Default value is 0. 0 No underflow 1 Underflow occurred
23 SMDU	Slow Serial Message DMA Underflow: Interrupt Status bit that indicates when there is an Underflow in Slow Serial Message DMA buffers on channel which has DMA enabled. The bit is set when there is no

Table continues on the next page...

SRX_GBL_STATUS field descriptions (continued)

Field	Description
	message in the DMA buffers for read and the software still reads the Slow Serial Message registers. Default value is 0. 0 No underflow 1 Underflow occurred
24–31 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

48.4.4 Fast Message Ready Status Register (SRX_FMSG_RDY)

The bits of this register are asserted when a Fast Message arrives on a corresponding channel that is not configured to be read via DMA. If a channel is not configured for DMA (via [Fast Message DMA Control Register \(SRX_FDMA_CTRL\)](#)), it is assumed to be read via Interrupt or Polling this status bit. Even if the Interrupt Enable bit for that channel is not set (via [Fast Message Ready Interrupt Control Register \(SRX_FRDY_IE\)](#)) and the channel is enabled, the message ready status bit in this register will be asserted on Fast Message reception. Software must ensure that if a channel is enabled; either DMA or Interrupt must be enabled for that channel. Proper flow cannot be guaranteed if neither is selected.

These bits can be configured to be cleared via "Write 1 to Clear" or "Fast Clearing Mechanism". The configuration is done by writing to the FAST_CLR bit in the [Global Control Register \(SRX_GBL_CTRL\)](#) .

NOTE

Reads of bits beyond those for the supported number of channels should be ignored. Writes to these bits is ignored by the Receiver.

Address: 0h base + Ch offset = Ch

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0																									F_RDY						
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

SRX_FMSG_RDY field descriptions

Field	Description
0–25 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
26–31 F_RDY	Fast Message Data Ready Status bits for channels 5 to 0 . Each bit indicates that a Fast Message is ready to be read by CPU from that specific channel. This bit is asserted only when DMA is NOT enabled for that

Table continues on the next page...

SRX_FMSG_RDY field descriptions (continued)

Field	Description
	channel (from the DMA Control Register) and an error free message is received, irrespective of the value of corresponding enable bit in Message Ready Interrupt Control Register. Reading the message from the corresponding memory-mapped Fast Message Read Registers would automatically clear this bit (Fast Clearing Mechanism).
	NOTE: These bits are either "write 1 to clear" (w1c) or self-clearing depending on the value of the FAST_CLR BIT in the Global Control Register (GBL_CTRL).
	The clearing mechanism is selected by user by programming the FAST_CLR bit in the Global Control register. See Section Global Control Register (SRX_GBL_CTRL) for details on FAST_CLR bit.
	NOTE: When using the fast clearing option, these bits will not clear till the complete message is read out which requires the CPU to read all 3 fast message registers via 32-bit read accesses. For 8/16-bit accesses fast clearing mechanism should not be used as it is not supported.
	NOTE: When the fast clearing option is not configured, the bits will be cleared only if 1 is written to those bits.
	0 No Fast Message is available to be read out via Interrupt
	1 Fast Message is available to be read out via Interrupt

48.4.5 Slow Serial Message Ready Status Register (SRX_SMSG_RDY)

The bits of this register are asserted when a Slow Serial Message arrives on a corresponding channel that is not configured to be read via DMA. If a channel is not configured for DMA (via [Fast Message DMA Control Register \(SRX_FDMA_CTRL\)](#) and [Slow Serial Message DMA Control Register \(SRX_SDMA_CTRL\)](#)), it is assumed to be read via Interrupt or Polling this status bit. Even if the Interrupt Enable bit for that channel is not set (via [Fast Message Ready Interrupt Control Register \(SRX_FRDY_IE\)](#) and [Slow Serial Message Ready Interrupt Enable Register \(SRX_SRDY_IE\)](#)) and the channel is enabled, the message ready status bit in this register will be asserted on Slow Message reception. Software must ensure that if a channel is enabled; either DMA or Interrupt must be enabled for that channel. Erroneous flow can happen if neither is selected.

These bits can be configured to be cleared via "Write 1 to Clear" or "Fast Clearing Mechanism". The configuration is done by writing to the FAST_CLR bit in the [Global Control Register \(SRX_GBL_CTRL\)](#).

NOTE

Reads of bits beyond those for the supported number of channels should be ignored. Writes to these bits is ignored by the Receiver.

Address: 0h base + 10h offset = 10h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0																									S_RDY						
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

SRX_SMSG_RDY field descriptions

Field	Description
0–25 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
26–31 S_RDY	Slow Serial Message Data Ready Status bits for channels 5 to 0. Each bit indicates that a Slow Serial Message is ready to be read by CPU from that specific channel. This bit is asserted only when DMA is NOT enabled for that channel (from the DMA Control Register) and an error free message is received, irrespective of the value of corresponding enable bit in Message Ready Interrupt Control Register. Reading the message from the corresponding memory-mapped Serial Message Read Registers would automatically clear this bit (Fast Clearing Mechanism). NOTE: These bits are either "write 1 to clear" (w1c) or self-clearing depending on the value of the FAST_CLR bit in the Global Control Register (GBL_CTRL). The clearing mechanism is selected by user by programming the FAST_CLR bit in the Global Control register. See Section Global Control Register (SRX_GBL_CTRL) for details on FAST_CLR bit. NOTE: When using the fast clearing option, these bits will not clear till the complete message is read out which requires the CPU to read all 3 Fast Message registers via 32-bit read accesses. For 8/16-bit accesses fast clearing mechanism should not be used as it is not supported. NOTE: When the fast clearing option is not configured, the bits will be cleared only if 1 is written to those bits. 0 No Slow Serial Message is available to be read out via Interrupt 1 Slow Serial Message is available to be read out via Interrupt

48.4.6 Data Control Register 1 (SRX_DATA_CTRL1)

The Data Control Register 1 (DATA_CTRL1) is described below.

NOTE

Reset value changes depending on the number of channels that are enabled on the device.

Address: 0h base + 18h offset = 18h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	NIBBCH0			0	NIBBCH1			0	NIBBCH2			0	NIBBCH3		
W																
Reset	0	0	0	1	0	0	0	1	0	0	0	1	0	0	0	1

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	NIBBCH4			0	NIBBCH5			0	Reserved			0	Reserved		
W																
Reset	0	0	0	1	0	0	0	1	0	0	0	0	0	0	0	0

SRX_DATA_CTRL1 field descriptions

Field	Description
0 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
1–3 NIBBCH0	Number of Data Nibbles supported in Channel 0 0 and 7 Invalid value. Must not be used 1 to 6 Number of data nibbles to be supported by respective channel
4 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
5–7 NIBBCH1	Number of Data Nibbles supported in Channel 1 0 and 7 Invalid value. Must not be used 1 to 6 Number of data nibbles to be supported by respective channel
8 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
9–11 NIBBCH2	Number of Data Nibbles supported in Channel 2 0 and 7 Invalid value. Must not be used 1 to 6 Number of data nibbles to be supported by respective channel
12 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
13–15 NIBBCH3	Number of Data Nibbles supported in Channel 3 0 and 7 Invalid value. Must not be used 1 to 6 Number of data nibbles to be supported by respective channel
16 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
17–19 NIBBCH4	Number of Data Nibbles supported in Channel 4 0 and 7 Invalid value. Must not be used 1 to 6 Number of data nibbles to be supported by respective channel
20 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
21–23 NIBBCH5	Number of Data Nibbles supported in Channel 5 0 and 7 Invalid value. Must not be used 1 to 6 Number of data nibbles to be supported by respective channel
24 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
25–27 Reserved	This field is reserved.
28 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
29–31 Reserved	This field is reserved.

48.4.7 Fast Message DMA Control Register (SRX_FDMA_CTRL)

NOTE

Reads of bits beyond those for the supported number of channels should be ignored. Writes to these bits is ignored by the Receiver.

RW

Address: 0h base + 28h offset = 28h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0																									FDMA_EN						
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

SRX_FDMA_CTRL field descriptions

Field	Description
0–25 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
26–31 FDMA_EN	Enable DMA for Fast Messages on channels 5 to 0 . These bits are writable when corresponding Fast Message Ready Interrupt Enable bits are set to 0. 0 DMA for Fast Messages is disabled 1 DMA for Fast Messages is enabled

48.4.8 Slow Serial Message DMA Control Register (SRX_SDMA_CTRL)

NOTE

Reads of bits beyond those for the supported number of channels should be ignored. Writes to these bits is ignored by the Receiver.

Address: 0h base + 2Ch offset = 2Ch

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0																									SDMA_EN						
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

SRX_SDMA_CTRL field descriptions

Field	Description
0–25 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
26–31 SDMA_EN	Enable DMA for Slow Serial Messages on channels 5 to 0 . These bits are writable when corresponding Slow Serial Message Ready Interrupt Enable bits are set to 0. 0 DMA for Slow Serial Messages is disabled 1 DMA for Slow Serial Messages is enabled

48.4.9 Fast Message Ready Interrupt Control Register (SRX_FRDY_IE)**NOTE**

Reads of bits beyond those for the supported number of channels should be ignored. Writes to these bits is ignored by the Receiver.

Address: 0h base + 34h offset = 34h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0																									FRDY_IE						
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

SRX_FRDY_IE field descriptions

Field	Description
0–25 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
26–31 FRDY_IE	Enable for Fast Message Ready Interrupt on channels 5 to 0 . These bits are writable when corresponding DMA enable bits are set to 0 in the Fast Message DMA Control Register. The availability of message is indicated via assertion of corresponding bit in Fast Message Ready Status Register irrespective of the value set in corresponding interrupt enable bit and DMA is not enabled. When a bit is asserted in this register, an interrupt for corresponding channel is generated. 0 Interrupt on reception of Fast Messages is disabled 1 Interrupt on reception of Fast Messages is enabled

48.4.10 Slow Serial Message Ready Interrupt Enable Register (SRX_SRDY_IE)

NOTE

Reads of bits beyond those for the supported number of channels should be ignored. Writes to these bits is ignored by the Receiver.

Address: 0h base + 38h offset = 38h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0																									SRDY_IE						
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

SRX_SRDY_IE field descriptions

Field	Description
0–25 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
26–31 SRDY_IE	Enable for Slow Serial Message Ready Interrupt on channels 5 to 0 . These bits are writable when corresponding DMA enable bits are set to 0 in the Slow Serial DMA Control Register. The availability of message is indicated via assertion of corresponding bit in Slow Serial Message Ready Status Register irrespective of the value set in corresponding interrupt enable bit and DMA is not enabled. When a bit in this register is asserted, an interrupt for corresponding channel is generated. 0 Interrupt on reception of Slow Serial Messages is disabled 1 Interrupt on reception of Slow Serial Messages is enabled

48.4.11 DMA Fast Message Data Read Register (SRX_DMA_FMSG_DATA)

DMA Source Size must be set to 32 bits when reading this register (DMA_FMSG_DATA) via DMA. When FIFO is enabled, the DMA must read the complete message using this register only. Three read accesses must be made to read complete message.

Address: 0h base + 40h offset = 40h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	CHNUM				SCNIB				DNIB1				DNIB2				DNIB3				DNIB4				DNIB5				DNIB6			
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

SRX_DMA_FMSG_DATA field descriptions

Field	Description
0–3 CHNUM	Channel Number. Indicates the channel number of the received message being read by DMA. Valid values are 0 to 5 .
4–7 SCNIB	Status and Communication Nibble of message
8–11 DNIB1	Data Nibble 1. Always supported
12–15 DNIB2	Data Nibble 2. Should be ignored if number of data nibbles supported by channel is 1
16–19 DNIB3	Data Nibble 3. Should be ignored if number of data nibbles supported by channel is 2 or less
20–23 DNIB4	Data Nibble 4. Should be ignored if number of data nibbles supported by channel is 3 or less
24–27 DNIB5	Data Nibble 5. Should be ignored if number of data nibbles supported by channel is 4 or less
28–31 DNIB6	Data Nibble 6. Should be ignored if number of data nibbles supported by channel is 5 or less

48.4.12 DMA Fast Message CRC Read Register (SRX_DMA_FMSG_CRC)

DMA Source Size must be set to 32 bits when reading this register (DMA_FMSG_CRC) via DMA. When FIFO is enabled, fast messages are read out from FIFO through the DMA Fast Message Data Read register only. Reading this register will generate an access error.

Address: 0h base + 44h offset = 44h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0												CRC4b				0															
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	

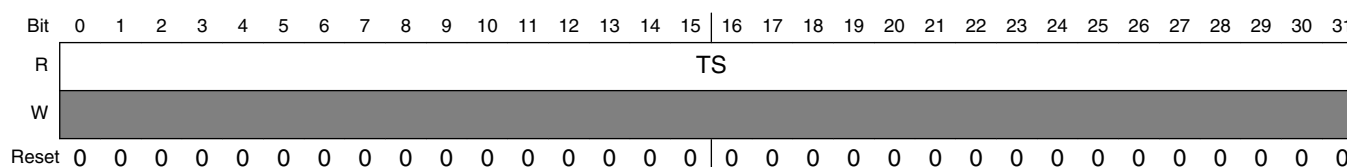
SRX_DMA_FMSG_CRC field descriptions

Field	Description
0–11 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
12–15 CRC4b	4-bit CRC value of the message.
16–31 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

48.4.13 DMA Fast Message Time Stamp Read Register (SRX_DMA_FMSG_TS)

The resolution of the time stamp is set by programming the Time Stamp Prescaler Value in the [Global Control Register \(SRX_GBL_CTRL\)](#) register. DMA Source Size must be set to 32 bits when reading this register (DMA_FMSG_TS) via DMA. When FIFO is enabled, fast messages are read out from FIFO through the DMA Fast Message Data Read register only. Reading this register will generate an access error.

Address: 0h base + 48h offset = 48h



SRX_DMA_FMSG_TS field descriptions

Field	Description
0–31 TS	Time Stamp for received message. Indicates the relative time when the falling edge that comes at the end of the calibration pulse (which is also the start of the status and communication nibble pulse) was detected

48.4.14 DMA Slow Serial Message Bit3 Read Register (SRX_DMA_SMSG_BIT3)

The above register (DMA_SMSG_BIT3) is common for all types of slow serial messages. Bit C defines the type of enhanced packet and bit TYPE defines whether it is short serial or enhanced serial message. Since this register is used to read both Short and Enhanced Serial Messages, the register fields have been placed in such a way to match the bit positions in actual message defined by SAE Specification as shown in [Table 48-83](#), [Table 48-84](#) and [Table 48-85](#). [Table 48-83](#) and [Table 48-84](#) shows the enhanced serial messages format to be used when reading this register for these two types of messages. [Table 48-85](#) shows the short serial message.

DMA Source Size must be set to 32 bits when reading this register via DMA.

NOTE

If the TYPE bit in the register reads '0', then remainder bits (for CFG, ID and DATA) will be redundant and read zeros. The short serial message will be located in DMA_SMSG_BIT2 register fields. If TYPE bit is '1', then the message in register is

Enhanced Serial Message and using CFG bit, software can decode the ID and DATA fields.

Address: 0h base + 50h offset = 50h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	CHNUM				TYPE	0										
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0				CFG		ID7_4_ID3_0				0	ID3_0_DATA15_12				0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

SRX_DMA_MSG_BIT3 field descriptions

Field	Description
0–3 CHNUM	Channel Number. Indicates the channel number of the received message being read by DMA. Valid values are 0 to 5 .
4 TYPE	Serial Message Type. Indicates the type of Serial Message received 0 Short Serial Message 1 Enhanced Serial Message
5–20 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
21 CFG	Configuration bit 'C'. Indicates the type of Enhanced Serial message 0 Enhanced Serial Message with 8-bit ID field 1 Enhanced Serial Message with 4-bit ID field
22–25 ID7_4_ID3_0	ID Field. Value depends on setting of 'C' bit. If 'C' bit is 0, this field is ID[7:4] and if the 'C' bit is 1, this field is ID[3:0]
26 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
27–30 ID3_0_DATA15_12	ID or Data Field. Value depends on setting of 'C' bit. If 'C' bit is 0, this field is ID[3:0] and if the 'C' bit is 1, this field is Data[15:12]
31 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

48.4.15 DMA Slow Serial Message Bit2 Read Register (SRX_DMA_SMSG_BIT2)

This register (DMA_SMSG_BIT2) is common for all types of slow serial messages. Hence, the register fields have been placed in such a way to match the bit positions in actual message defined by SAE Specification as shown in [Table 48-83](#), [Table 48-84](#) and [Table 48-85](#). [Table 48-83](#), and [Table 48-84](#) shows the enhanced serial messages format to be used when reading this register for these two types of messages. [Table 48-85](#) shows the short serial message.

DMA Source Size must be set to 32 bits when reading this register via DMA.

Address: 0h base + 54h offset = 54h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	
R	0										SMCRC						0				DATA[11:0]												
W																																	
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	

SRX_DMA_SMSG_BIT2 field descriptions

Field	Description
0–9 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
10–15 SMCRC	6-bit CRC value of the message. When TYPE bit in DMA Slow Serial Message Bit3 Read Register (SRX_DMA_SMSG_BIT3) is set to 0, this CRC is of 4-bits for Short Serial Messages and when TYPE bit is set to 1, this CRC is of 6-bits for Enhanced Serial Messages.
16–19 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
20–31 DATA[11:0]	12-bit Data Value

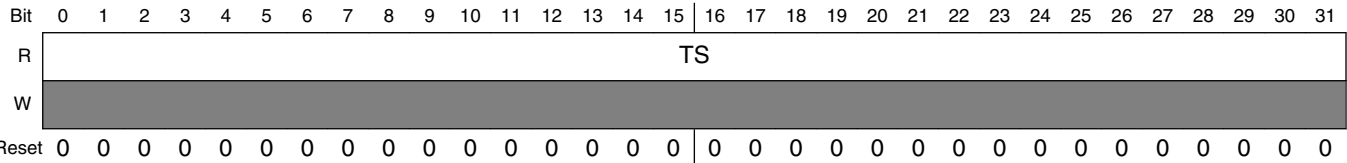
48.4.16 DMA Slow Serial Message Time Stamp Read Register (SRX_DMA_SMSG_TS)

The resolution of the time stamp is set by programming the Time Stamp Prescaler Value in the [Global Control Register \(SRX_GBL_CTRL\)](#) register.

DMA Source Size must be set to 32 bits when reading this register (DMA_SMSG_TS) via DMA.

Memory map and register definition

Address: 0h base + 58h offset = 58h



SRX_DMA_SMSG_TS field descriptions

Field	Description
0–31 TS	Time Stamp for the received message. Indicates the relative time of detection of the falling edge that comes at the end of the calibration pulse (which is also the start of the status and communication nibble pulse) of the last Fast Message that completed the Serial Message. Time Stamp for Slow Serial Messages is not latched when the Slow Serial Message starts but when the Slow Serial Message is completely received.

48.4.17 Channel 'n' Clock Control Register (n = 0 to (CH-1)) (SRX_CHn_CLK_CTRL)

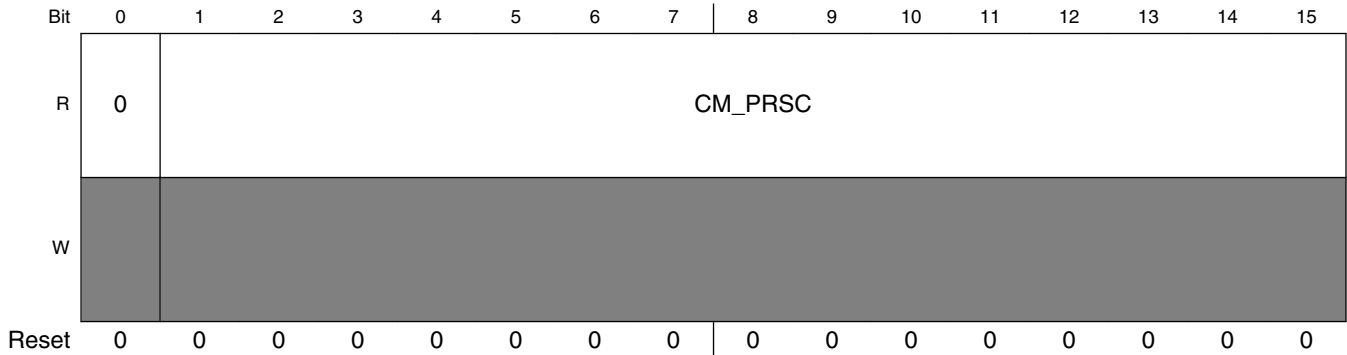
NOTE

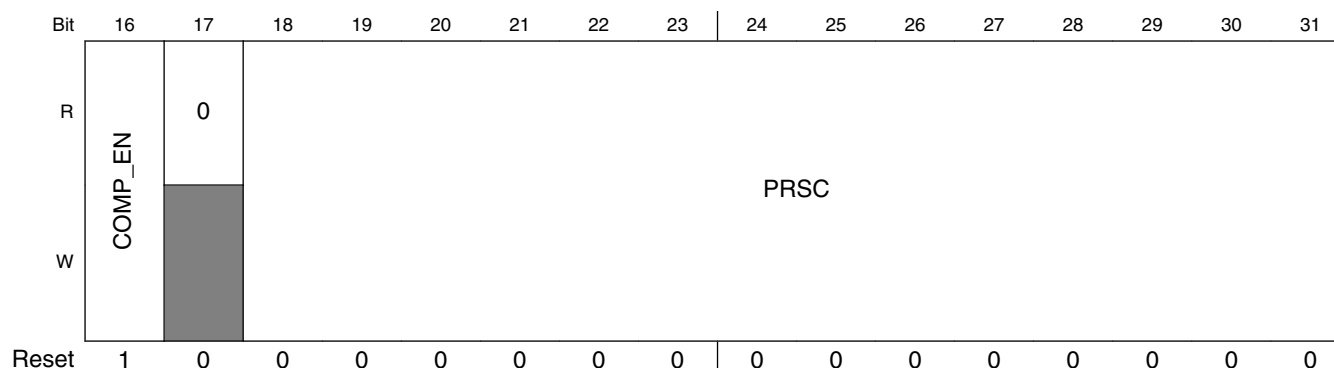
The value of the CH is device-specific. See the device configuration section for details.

The SENT Receiver module generates the Receiver Sensor Clock (or simply Receiver Clock) internally to sample the incoming filtered sensor data. The user software must program the prescaler value (for Receiver Clock) to match the ideal Sensor Tx Clock period. The generated Receiver Clock is compensated for variations and clock drift that might occur in the Sensor Tx Clock. See [Adjustment for variation in sensor \(Tx\) clock](#) for more details.

This register configures the generation and subsequent compensation logic of the Receiver Clock. User must program the contents of this register before enabling the channel by writing to the corresponding channel enable bit in the [Channel Enable Register \(SRX_CHNL_EN\)](#).

Address: 0h base + 60h offset + (16d × i), where i=0d to 5d





SRX_CHn_CLK_CTRL field descriptions

Field	Description
0 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
1–15 CM_PRSC	Compensated Prescaler Value. This is the compensated prescaler value calculated by Clock Compensation logic and this value is used to compensate for sensor Tx clock variation. This value is indicated as a read only field for use by user software, if needed. This is a 15-bit value. '= Rx Prescaler Value' No variation, hence no compensation or Compensation is disabled '< Rx Prescaler Value' Tx Clock is SLOWER than the programmed receive clock '> Rx Prescaler Value' Tx Clock is FASTER than the programmed receive clock Thus, the new Receiver Clock Tick Period can be computed by software as Compensated Prescaler Value x High Frequency Receiver Clock Period Note: The value provided in this field is ceil value of actual tick period as measured by the Receiver.
16 COMP_EN	Compensation Enable. To enable compensation logic to adjust the receiver clock against variation in Tx clock on this channel. User must set the Prescaler Value before enabling this bit. 0 Compensation is disabled 1 Compensation is enabled
17 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
18–31 PRSC	Rx Prescaler Value. This factor is required to generate required Receiver Clock tick from the High Frequency Receiver Clock. This value can only be written when Channel Enable bit is Zero. Hence this value must be programmed before enabling the channel. If required receiver clock tick period (Trx_clk) is not divisible by the high frequency receiver clock (protocol clock) period (Thf_clk), the required prescaler value will have a fractional part which cannot be programmed. In this case then user is advised to program the prescaler value to the lower integer value. 0 Prescaler is bypassed. Should not be used. Non-Zero Value Enables Prescaler The required value is computed by software as follows: $\text{Prescaler\_value} = \text{floor}\left(\frac{T_{\text{rx\_clk}}}{T_{\text{hf\_clk}}}\right)$ (both clock periods taken in μs). Please see High frequency receiver clock (protocol clock) requirements before setting this value.

Table continues on the next page...

SRX_CHn_CLK_CTRL field descriptions (continued)

Field	Description
	e.g. if the required receiver clock tick period is 43 μ s and the High Frequency Receiver clock is selected to be 62.5 MHz (or 0.016 μ s) then, Prescaler Value = $43/0.016 = 2687.5$. Hence user must program 2687 in this register.

**48.4.18 Channel 'n' Status Register (n=0 to (CH-1))
(SRX_CHn_STATUS)**

This registers stores information about certain events and error conditions that occur in the channel during the message reception process. The assertion of these bit is not linked to any message but it is a general alert to the CPU that a particular condition (error or normal event) happened in the recent past. Messages are automatically discarded when an error is detected.

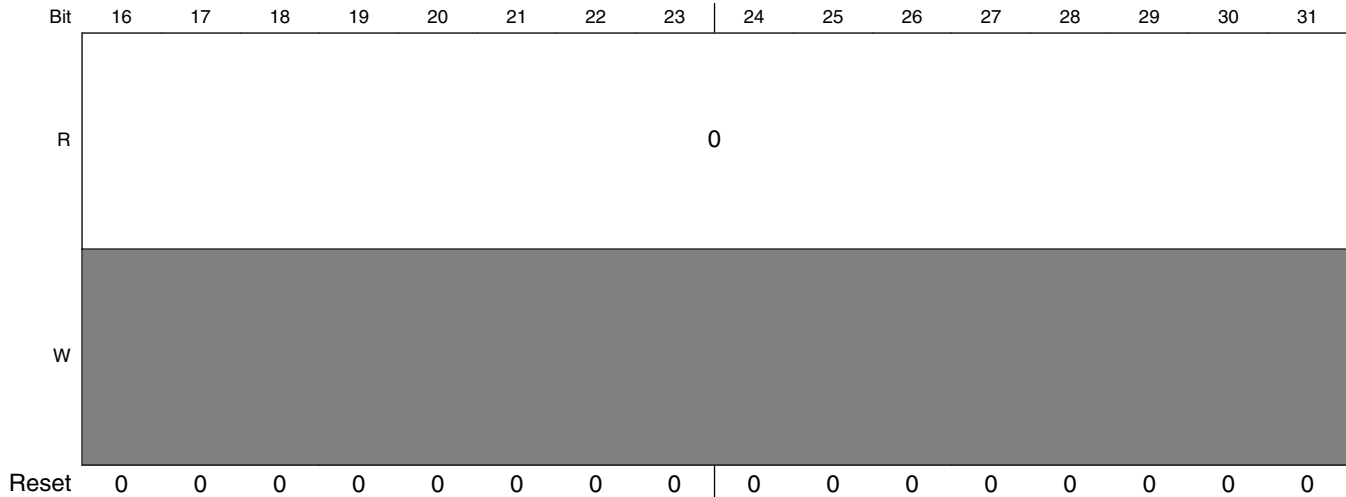
User software can clear these bit at any time by writing 1 to them. This register can be updated at any time irrespective of the setting in the [Channel Enable Register \(SRX_CHNL_EN\)](#) bit for the corresponding channel.

NOTE

The value of the CH is device-specific. See the device configuration section for details.

Address: 0h base + 64h offset + (16d $\times$ i), where i=0d to 5d

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	BUS_IDLE	0			CAL_RESYNC	CAL_20_25	SMSG_OFLW	FMSG_OFLW	0	PP_DIAG_ERR	CAL_LEN_ERR	CAL_DIAG_ERR	NIB_VAL_ERR	SMSG_CRC_ERR	FMSG_CRC_ERR	NUM_EDGES_ERR
W	w1c				w1c	w1c	w1c	w1c		w1c	w1c	w1c	w1c	w1c	w1c	w1c
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0



SRX_CHn_STATUS field descriptions

Field	Description
0 BUS_IDLE	Bus Idle Status. This bit indicates that the sensor interface has been idle for more than the period defined by the programmed Bus Idle Count value(in Channel 'n' Configuration Register (n=0 to (CH-1)) (SRX_CHn_CONFIG)). CPU should write 1 to this clear this bit once it is read. 0 Bus is not idle 1 Channel has been idle for more than the allowed value
1-3 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
4 CAL_RESYNC	Successive Calibration Check (Option 2 of SAE SENT Spec) Resynchronized. This bit indicates that Successive Calibration Check (Option 2 of SAE Spec) has failed three times and has resynchronized to assume 3rd message to be correct and will be received. The successive calibration check error will also be asserted during re-synchronization (to indicate the 3rd error being detected). 0 No interrupt 1 Interrupt Status condition has occurred
5 CAL_20_25	Calibration Variation 20 - 25% Interrupt Status. This bit indicates that the calibration pulse received on this channel has variation in between $\pm 20\%$ to $\pm 25\%$ from 56 ticks. 0 No Interrupt 1 Interrupt Status condition has occurred
6 SMSG_OFLW	Slow Serial Message Overflow Status. This bit indicates overflow occurred on this channel. Overflow will cause state machine to halt stopping reception of new messages. 0 No Interrupt 1 Interrupt Status condition has occurred
7 FMSG_OFLW	Fast Message Overflow Status. This bit indicates overflow occurred on this channel. Overflow happens when effective data read rate by CPU is slower than data receive rate on channel. On overflow, new incoming messages will overwrite the previous messages (that are not read out) stored in the receiver buffers. However, any message that is currently being read out by DMA or CPU will not get over-written. 0 No Interrupt 1 Interrupt Status condition has occurred

Table continues on the next page...

SRX_CHn_STATUS field descriptions (continued)

Field	Description
8 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
9 PP_DIAG_ERR	This diagnostic checks status bit indicates that the ratio of calibration pulse length to overall message length (with pause pulse) is more than $\pm 1.5625\%$ between two messages. This check is valid only for messages with pause pulse. This bit indicates whether this diagnostic has failed or passed on this channel, and is cleared by writing a 1 to it. 0 Error Check has passed 1 Error Check has failed
10 CAL_LEN_ERR	This diagnostic checks status bit indicates that Calibration pulse is more than 56 ticks $\pm 25\%$. This bit indicates whether this diagnostic has failed or passed on this channel, and is cleared by writing a 1 to it. 0 Error Check has passed 1 Error Check has failed
11 CAL_DIAG_ERR	Successive Calibration pulses differ by $\pm 1.56\%$. SAE spec defines it be 1.5625% but the accuracy of this check depends on the frequency of protocol clock (See High frequency receiver clock (protocol clock) requirements). This diagnostic checks status bit indicates whether this diagnostic has failed or passed on this channel, and is cleared by writing a 1 to it. 0 Error Check has passed 1 Error Check has failed
12 NIB_VAL_ERR	Any nibble data value <0 or >15 . This diagnostic checks status bit indicates whether this diagnostic has failed or passed on this channel, and is cleared by writing a 1 to it. 0 Error Check has passed 1 Error Check has failed
13 SMSG_CRC_ERR	Checksum error in Slow Serial Message. This diagnostic checks status bit indicates whether this diagnostic has failed or passed on this channel, and is cleared by writing a 1 to it. 0 Error Check has passed 1 Error Check has failed
14 FMSG_CRC_ERR	Checksum error in Fast Message. This diagnostic checks status bit indicates whether this diagnostic has failed or passed on this channel, and is cleared by writing a 1 to it. 0 Error Check has passed 1 Error Check has failed
15 NUM_EDGES_ERR	Not the expected number of negative edges between calibration pulse. This diagnostic checks status bit indicates whether this diagnostic has failed or passed on this channel. This bit will not set in case of "Option 2" for successive calibration pulse check is selected. This error will only set when "Option 1" for successive calibration check is selected (SUCC_CAL_CHK). This bit is cleared by writing a 1 to it. 0 Error Check has passed 1 Error Check has failed
16–31 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

48.4.19 Channel 'n' Configuration Register (n=0 to (CH-1)) (SRX_CHn_CONFIG)

This register (CHn_CONFIG) provides the configurability for interrupt assertion for various events and error as explained in [Channel 'n' Status Register \(n=0 to \(CH-1\)\) \(SRX_CHn_STATUS\)](#) and control bits for selection of various receive options that might be required by the user software. This register also configures the width of noise glitches (on the sensor input) to be filtered out by the input programmable filter.

User must program the contents of this register before enabling the channel by writing to the corresponding channel enable bit in the [Channel Enable Register \(SRX_CHNL_EN\)](#).

NOTE

The value of the CH is device-specific. See the device configuration section for details.

Address: 0h base + 68h offset + (16d × i), where i=0d to 5d

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	BUS_IDLE_CNT				IE_CAL_RESYNC	IE_CAL_20_25	IE_SMSG_OFLW	IE_FMSG_OFLW	FCRC_CHK_OFF	IE_PP_DIAG_ERR	IE_CAL_LEN_ERR	IE_CAL_DIAG_ERR	IE_NIB_VAL_ERR	IE_SMSG_CRC_ERR	IE_FMSG_CRC_ERR	IE_NUM_EDGES_ERR
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	DCHNG_INT	CAL_RNG	PP_CHKSEL	FCRC_TYPE	FCRC_SC_EN	SCRC_TYPE	PAUSE_EN	SUCC_CAL_CHK	FIL_CNT							
W																
Reset	0	0	0	0	0	0	0	1	0	0	0	0	0	1	0	0

SRX_CHn_CONFIG field descriptions

Field	Description
0–3 BUS_IDLE_CNT	Bus Idle Count. This register value defines the maximum allowable idle period on the sensor interface of this channel. The value is defined as follows: 0000 Disabled. 0001 127*2 Receiver Clock Tick Counts 0010 127*4 Receiver Clock Tick Counts 0100 127*8 Receiver Clock Tick Counts 1000 127*16 Receiver Clock Tick Counts

Table continues on the next page...

SRX_CHn_CONFIG field descriptions (continued)

Field	Description
4 IE_CAL_ RESYNC	Successive Calibration Check Resynchronized Interrupt Enable: This bit enables interrupt assertion when Successive Calibration diagnosis has failed three times in case of "Option 2" being selected for Successive Calibration Check Method and the check resynchronizes the take the third message to be correct. 0 Interrupt is disabled 1 Interrupt is enabled
5 IE_CAL_20_25	Calibration Variation 20 - 25% Interrupt Enable. This bit enables interrupt assertion when corresponding status bit is set. 0 Interrupt is disabled 1 Interrupt is enabled
6 IE_SMSG_OFLW	Slow Serial Message Overflow Interrupt Enable. This bit enables interrupt assertion when overflow occurs on reception of Slow Serial Messages in corresponding channel 0 Interrupt is disabled 1 Interrupt is enabled
7 IE_FMSG_OFLW	Fast Message Overflow Interrupt Enable. This bit enables interrupt assertion when overflow occurs on reception of Fast Messages in corresponding channel. 0 Interrupt is disabled 1 Interrupt is enabled
8 FCRC_CHK_ OFF	Fast Message CRC Check Off: This bit can be used to switch off CRC check in Fast Message 0 Check is enabled 1 Check is disabled/off
9 IE_PP_DIAG_ ERR	Ratio of calibration pulse length to message length varies by more than $\pm 1.5625\%$ between two frames. Valid for messages with pause pulse. This bit enables interrupt assertion when its respective diagnostic check fails on the corresponding channel. 0 Interrupt is disabled 1 Interrupt is enabled
10 IE_CAL_LEN_ ERR	Calibration pulse is wider than 56 ticks $\pm 25\%$. This bit enables interrupt assertion when its respective diagnostic check fails on the corresponding channel. 0 Interrupt is disabled 1 Interrupt is enabled
11 IE_CAL_DIAG_ ERR	Successive Calibration pulses differ by more than $\pm 1.56\%$. SAE spec defines it be 1.5625% but the accuracy of this check depends on the protocol clock. For instance for 62.5 MHz, this check will fail for 1.56% and passes for 1.55%. This bit enables interrupt assertion when its respective diagnostic check fails on the corresponding channel. 0 Interrupt is disabled 1 Interrupt is enabled
12 IE_NIB_VAL_ ERR	Any nibble data value <0 or >15 . This bit enables interrupt assertion when its respective diagnostic check fails on the corresponding channel. 0 Interrupt is disabled 1 Interrupt is enabled

Table continues on the next page...

SRX_CHn_CONFIG field descriptions (continued)

Field	Description
13 IE_SMSG_CRC_ERR	Checksum error in Slow Serial Message. This bit enables interrupt assertion when its respective diagnostic check fails on the corresponding channel. 0 Interrupt is disabled 1 Interrupt is enabled
14 IE_FMSG_CRC_ERR	Checksum error in Fast Message. This bit enables interrupt assertion when its respective diagnostic check fails on the corresponding channel. 0 Interrupt is disabled 1 Interrupt is enabled
15 IE_NUM_EDGES_ERR	Not the expected number of negative edges between calibration pulse. This bit enables interrupt assertion when its respective diagnostic check fails on the corresponding channel. 0 Interrupt is disabled 1 Interrupt is enabled
16 DCHNG_INT	Enable for Interrupt on Reception of Fast Message with Changed Data: This bit is used to enable storing and generating corresponding DMA or Message Ready Interrupt Request for Fast Message only when any of the data nibbles after different from previously received Fast Message on that channel. If a new Fast Message is received that has same values for all data nibbles as that of previous message then the current message will be discarded and will not be stored in the buffer, or FIFO, and neither Fast Message Ready Interrupt nor DMA request will be generated. 0 All Fast Messages will be received even if data nibbles are same 1 Only Fast Messages with differing values of data nibbles will be received
17 CAL_RNG	Valid Calibration Pulse Range Selection: This bit is used to control whether 20% variation or 25% variation in Calibration Pulse will be tolerated when detecting a Calibration Pulse 0 20% variation is acceptable 1 25% variation is acceptable
18 PP_CHKSEL	Pause Pulse Diagnostic Check Selection: This bit controls which diagnostic checks to run when messages are enabled to be received with pause pulses. 0 Both Successive Calibration Pulse Check and Pause Pulse Diagnostic are run 1 Only Pause Pulse Diagnostic is run. Successive Calibration Pulse Check is disabled.
19 FCRC_TYPE	Fast Message CRC Type. This indicates the type of CRC method to be used for CRC Diagnostic Check on the received fast messages 0 Recommended implementation (additional 0 data nibble XORed with the rest of the nibbles) 1 Legacy implementation (no additional 0 data nibble is XORed)
20 FCRC_SC_EN	Fast Message CRC Status and Communication Nibble Enable. This bit enables the inclusion of Status and Communication Nibble when CRC is calculated for Fast Messages. 0 Status and Communication Nibble not included in CRC calculation 1 Status and Communication Nibble is included in CRC calculation
21 SCRC_TYPE	Slow Serial Message CRC Type. This indicates the type of CRC method to be used for CRC Diagnostic Check on the received slow serial messages. 0 Recommended implementation (additional 0 data nibble XORed with the rest of the nibbles) 1 Legacy implementation (no additional 0 data nibble XORed)
22 PAUSE_EN	Pause Pulse Enable. Enables the channel receiver to detect a pause pulse.

Table continues on the next page...

SRX_CHn_CONFIG field descriptions (continued)

Field	Description
	0 Detection of Pause Pulse is disabled 1 Detection of Pause Pulse enabled
23 SUCC_CAL_CHK	Successive Calibration Pulse Check Method. This bit indicates the method to be used for performing the successive calibration pulse check. Default value is 1. 0 Option 2 i.e. Low Latency Option as per SAE Specification 1 Option 1 i.e. Preferred but High Latency Option as per SAE Specification
24–31 FIL_CNT	Input Filter Sample Count. This indicates twice the number of Protocol Clock (of High Frequency Receiver Clock) cycles required for channel input from device's pad to remain stable before it is sampled as 0 or 1. This field defines the width of glitches to be filtered out by the input programmable filter on that channel. Only one bit should be set in this field and the output of filter will be offset by twice the same number of clocks plus additional 3 clocks due to synchronization. Default value is '4'. 0 No filtering. Input from device's pad is only synchronized 1 Non-Zero Filtering is enabled

48.4.20 Channel 'n' Fast Message Data Read Register (n=0 to (CH-1)) (SRX_CHn_FMSG_DATA)

Address offset: 0x0160 + 0x18 x n

This is register CHn_FMSG_DATA. The contents of this register are same as [DMA Fast Message Data Read Register \(SRX_DMA_FMSG_DATA\)](#) .

NOTE

The value of the CH is device-specific. See the device configuration section for details.

Address: 0h base + 160h offset + (24d × i), where i=0d to 5d

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	CHNUM				SCNIB				DNIB1				DNIB2				DNIB3				DNIB4				DNIB5				DNIB6			
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	

SRX_CHn_FMSG_DATA field descriptions

Field	Description
0–3 CHNUM	Channel Number. Indicates the channel number of the received message being read by DMA. Valid values are 0 to 5 .
4–7 SCNIB	Status and Communication Nibble of message
8–11 DNIB1	Data Nibble 1. Always supported

Table continues on the next page...

SRX_CHn_FMSG_DATA field descriptions (continued)

Field	Description
12–15 DNIB2	Data Nibble 2. Should be ignored if number of data nibbles supported by channel is 1
16–19 DNIB3	Data Nibble 3. Should be ignored if number of data nibbles supported by channel is 2 or less
20–23 DNIB4	Data Nibble 4. Should be ignored if number of data nibbles supported by channel is 3 or less
24–27 DNIB5	Data Nibble 5. Should be ignored if number of data nibbles supported by channel is 4 or less
28–31 DNIB6	Data Nibble 6. Should be ignored if number of data nibbles supported by channel is 5 or less

48.4.21 Channel 'n' Fast Message CRC Read Register (n=0 to (CH-1)) (SRX_CHn_FMSG_CRC)

Address offset: $0x0164 + 0x18 \times n$

This is register CHn_FMSG_CRC. The contents of this register are same as [DMA Fast Message CRC Read Register \(SRX_DMA_FMSG_CRC\)](#).

Address: $0h \text{ base} + 164h \text{ offset} + (24d \times i)$, where $i=0d$ to $5d$

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0												CRC4b				0															
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	

SRX_CHn_FMSG_CRC field descriptions

Field	Description
0–11 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
12–15 CRC4b	4-bit CRC value of the message.
16–31 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

48.4.22 Channel 'n' Fast Message Time Stamp Read Register (n=0 to (CH-1)) (SRX_CHn_FMSG_TS)

Address offset: $0x0168 + 0x18 \times n$

This is register CHn_FMSG_TS. The contents of this register are same as [DMA Fast Message Time Stamp Read Register \(SRX_DMA_FMSG_TS\)](#).

Memory map and register definition

Address: 0h base + 168h offset + (24d × i), where i=0d to 5d

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R																TS																
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

SRX_CHn_FMSG_TS field descriptions

Field	Description
0–31 TS	Time Stamp for received message. Indicates the relative time when the falling edge that comes at the end of the calibration pulse (which is also the start of the status and communication nibble pulse) was detected

48.4.23 Channel 'n' Serial Message Read Register (Bit 3) (n=0 to (CH-1)) (SRX_CHn_SMSG_BIT3)

Address offset: 0x016C + 0x18 × n

This is register CHn_SMSG_BIT3. The contents of this register are same as [DMA Slow Serial Message Bit3 Read Register \(SRX_DMA_SMSG_BIT3\)](#) .

NOTE

The value of the CH is device-specific. See the device configuration section for details.

Address: 0h base + 16Ch offset + (24d × i), where i=0d to 5d

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	CHNUM				TYPE	0										
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0				CFG	ID7_4_ID3_0				0	ID3_0_DATA15_12				0	
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

SRX_CHn_SMSG_BIT3 field descriptions

Field	Description
0–3 CHNUM	Channel Number. Indicates the channel number of the received message being read by DMA. Valid values are 0 to 5 .
4 TYPE	Serial Message Type. Indicates the type of Serial Message received 0 Short Serial Message 1 Enhanced Serial Message

Table continues on the next page...

SRX_CHn_SMSG_BIT3 field descriptions (continued)

Field	Description
5–20 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
21 CFG	Configuration bit 'C'. Indicates the type of Enhanced Serial message 0 Enhanced Serial Message with 8-bit ID field 1 Enhanced Serial Message with 4-bit ID field
22–25 ID7_4_ID3_0	ID Field. Value depends on setting of 'C' bit. If 'C' bit is 0, this field is ID[7:4] and if the 'C' bit is 1, this field is ID[3:0]
26 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
27–30 ID3_0_DATA15_12	ID or Data Field. Value depends on setting of 'C' bit. If 'C' bit is 0, this field is ID[3:0] and if the 'C' bit is 1, this field is Data[15:12]
31 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

48.4.24 Channel 'n' Serial Message Read Register (Bit 2)(n=0 to (CH-1)) (SRX_CHn_SMSG_BIT2)

Address offset: $0x0170 + 0x18 \times n$

This is register CHn_SMSG_BIT2. The contents of this register are same as [DMA Slow Serial Message Bit2 Read Register \(SRX_DMA_SMSG_BIT2\)](#) .

Address: $0h \text{ base} + 170h \text{ offset} + (24d \times i)$, where $i=0d$ to $5d$

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0										SMCRC						0			DATA[11:0]												
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	

SRX_CHn_SMSG_BIT2 field descriptions

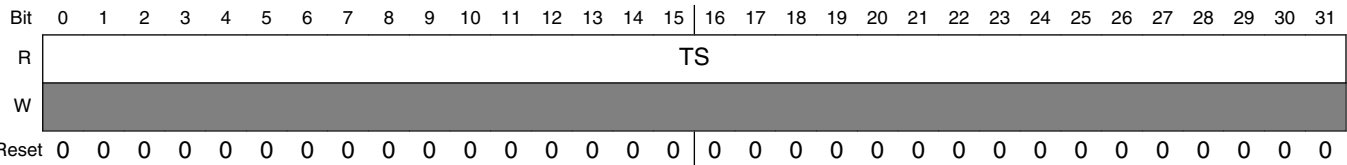
Field	Description
0–9 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
10–15 SMCRC	6-bit CRC value of the message. When TYPE bit in DMA Slow Serial Message Bit3 Read Register (SRX_DMA_SMSG_BIT3) is set to 0, this CRC is of 4-bits for Short Serial Messages and when TYPE bit is set to 1, this CRC is of 6-bits for Enhanced Serial Messages.
16–19 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
20–31 DATA[11:0]	12-bit Data Value

48.4.25 Channel 'n' Serial Message Time Stamp Read Register (n=0 to (CH-1)) (SRX_CHn_SMSG_TS)

Address offset: 0x0174 + 0x18 x n

This is register CHn_SMSG_TS. The contents of this register are same as [DMA Slow Serial Message Time Stamp Read Register \(SRX_DMA_SMSG_TS\)](#).

Address: 0h base + 174h offset + (24d × i), where i=0d to 5d



SRX_CHn_SMSG_TS field descriptions

Field	Description
0–31 TS	Time Stamp for the received message. Indicates the relative time of detection of the falling edge that comes at the end of the calibration pulse (which is also the start of the status and communication nibble pulse) of the last Fast Message that completed the Serial Message. Time Stamp for Slow Serial Messages is not latched when the Slow Serial Message starts but when the Slow Serial Message is completely received.

48.5 Functional description

This section describes the function of the SRX.

48.5.1 Initialization sequence

All the channels are required to be configured before enabling them by writing into corresponding channel enable bit in the [Channel Enable Register \(SRX_CHNL_EN\)](#). There are some registers which must be configured and some which are to be changed if reset value is not as required for that channel. After enabling the channel, no configuration should be changed and if required to be changed, the user needs to first disable the channel and then reconfigure the channel parameters. The following must to be configured before enabling the corresponding channel:

- Time Stamp Prescaler Value in [Global Control Register \(SRX_GBL_CTRL\)](#) to generate the time stamping clock

- Rx Prescaler Value in [Channel 'n' Clock Control Register \(n = 0 to \(CH-1\)\) \(SRX_CHn_CLK_CTRL\)](#) to generate the channel's receiver clock
- Compensation enable in [Channel 'n' Clock Control Register \(n = 0 to \(CH-1\)\) \(SRX_CHn_CLK_CTRL\)](#)
- Channel Configuration settings in the [Channel 'n' Configuration Register \(n=0 to \(CH-1\)\) \(SRX_CHn_CONFIG\)](#)
- Number of Data Nibbles in Channel in the [Data Control Register 1 \(SRX_DATA_CTRL1\)](#)
- Control bits that control whether messages received in channel are to read by interrupt or DMA in [Fast Message DMA Control Register \(SRX_FDMA_CTRL\)](#), [Slow Serial Message DMA Control Register \(SRX_SDMA_CTRL\)](#), [Fast Message Ready Interrupt Control Register \(SRX_FRDY_IE\)](#) and [Slow Serial Message DMA Control Register \(SRX_SDMA_CTRL\)](#)

48.5.2 DMA read logic

The SENT Receiver module generates separate DMA requests for all Fast Messages and Slow Serial Messages received. The DMA requests are common for all channels. Each packet stored in the channels' buffers are appended with a time stamp taken from a free running time keeping counter and the channel number on which it was received. Details on this time stamp counter can be found in [Time stamp logic](#).

48.5.2.1 DMA request for Fast Message reading via FIFO

A single DMA request is generated for all Fast Messages received on any channel. There is a DMA read FIFO to store the incoming messages from all the channels and allow them to be read via DMA. The DMA operation will happen as follows:

- DMA request will be asserted when number of messages stored in FIFO crosses the configured watermark level (in Global Control Register (GBL_CTRL)).
- DMA request will be de-asserted when number of messages stored goes below watermark level. It is assumed that DMA Controller's TCD would be set to transfer bytes equal to the messages indicated by watermark level so as to empty the FIFO.
- It is assumed that when responding to a DMA request, the DMA controller will read at a rate faster than the rate at which messages are getting received and stored in FIFO.

- The complete received message can be read by three consecutive read accesses to DMA Fast Message Data Read Register. The message will be read out in following sequence:
 - Fast Message data as described by DMA Fast Message Data Read Register (DMA_FMSG_DATA)
 - CRC for the Fast Message as described by DMA Fast Message CRC Read Register (DMA_FMSG_CRC)
 - Time stamp value for the Fast Message as described by DMA Fast Message Time Stamp Read Register (DMA_FMSG_TS)
- The channel number is prefixed in each valid message (no error) to allow software to identify messages pertaining to a channel
- [Table 48-82](#) shows the format in which the messages will be stored in any memory-mapped message buffer and when being read out from FIFO.
- Since multiple messages can be received at the same time, there is a [Fixed round robin sequencing logic](#), which will scan all active channels configured for DMA and store their message into the FIFO one by one. Channel configured for DMA but do not have ready messages will be automatically skipped till their turn comes again in the round robin sequence.
- Messages received can be stored in a different order to which they are actually received. The time stamp will indicate the order of reception of messages in a particular channel. However, the time stamp cannot be used to determine the order across channels.

This figure below shows the data for the three 32-bit accesses to be made by DMA to get one complete message with time stamp.

Table 48-82. Format of storing of Fast Messages in FIFO

CH NUM [3:0]	Status and Comm [3:0]	Data 1 [3:0]	Data 2 [3:0]	Data 3 [3:0]	Data 4 [3:0]	Data 5 [3:0]	Data 6 [3:0]
			CRC [3:0]				
TIME STAMP [31:0]							

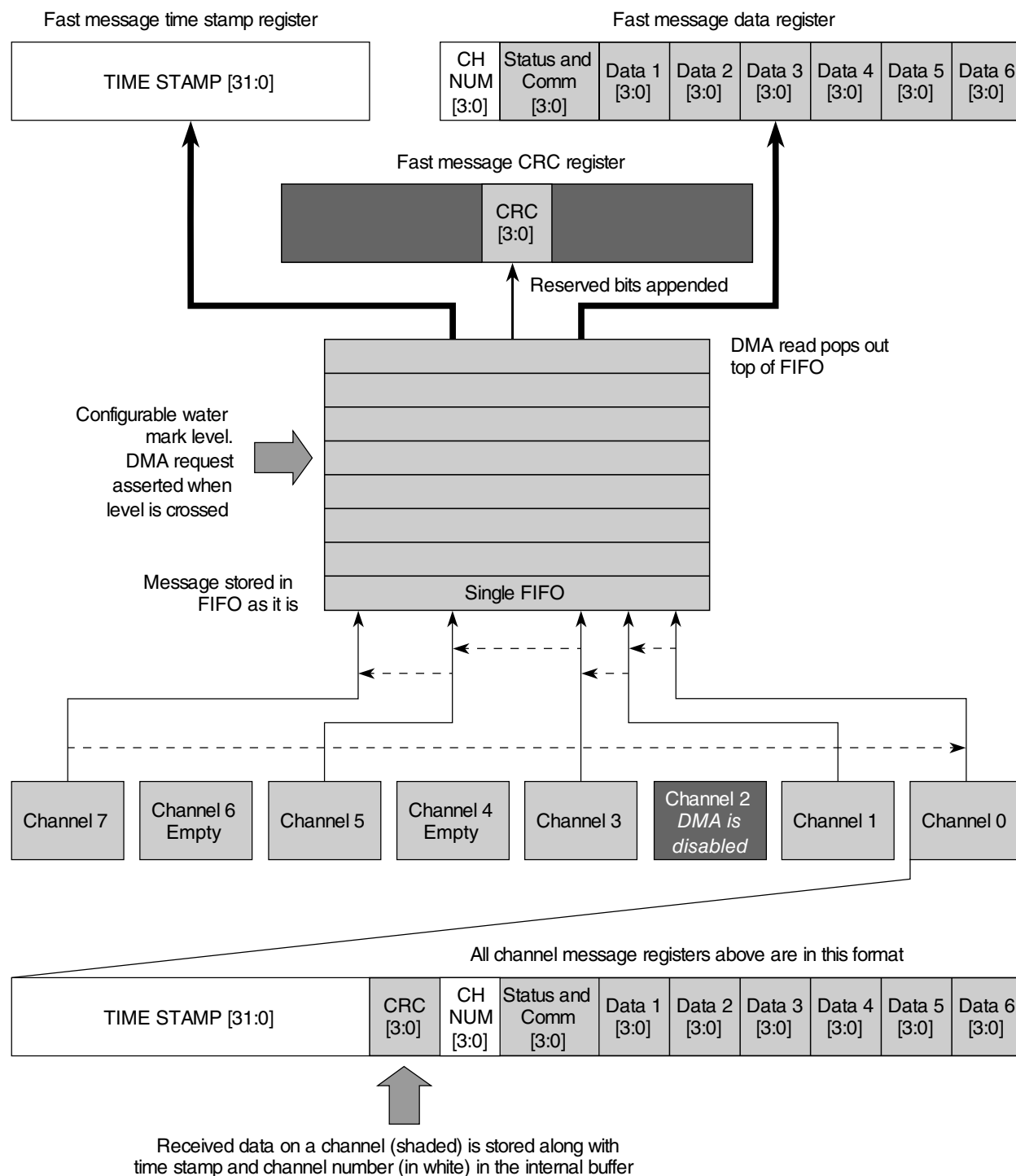


Figure 48-81. Fast Message DMA Read Logic using FIFO

48.5.2.1.1 Fixed round robin sequencing logic

Figure 48-81 shows a scenario where 8 channels (0-7) with all channels are enabled for DMA reading except channel 2. DMA request will be asserted when FIFO has received packets more than watermark level set. Figure 48-81 shows channels 6 and 4 have not received the complete message. The following points describe the complete round robin sequencing scheme.

- The round robin sequencing logic which stores the messages in the FIFO by looping through each channels' Fast Message register and checking for the availability of the message.
- As shown in Figure 48-81, the arbitration logic will check for messages in sequence 0 -> 1 -> 3 -> 5 -> 7 and back to channel 0 and keep looping. Channel 2 is skipped as it's not enabled to be read via DMA) and channel 4 and 6 are skipped as message is not received completely at the instance the sequencing logic checked their buffers. This sequence continues in this fashion.
- The sequence in which messages are stored in FIFO is not deterministic and depends on the sequence they are detected after the noise filtering by the input filter of the channel.
- The DMA controller responds to the DMA request by reading from the DMA Fast Message Read register. When a DMA read happens, a message is read from the FIFO and split into three 32-bit values and provided to DMA in the order as described in previous section. The three 32-bit values together form one complete Fast Message.
- Once all three 32-bit access are completed, the next data from the FIFO is popped.

Figure 48-81 also shows how the received data (light gray colored boxes) from a channel is stored in FIFO along with the time stamp and channel number appended (both white colored).

48.5.2.2 DMA request for Fast Message reading when FIFO is disabled

In this mode no FIFO will be used as it is disabled via the FIFO Enable bit in the Global Control Register (GBL_CTRL). Fast Messages received are stored in a single system bus clock domain buffer to be read out by the DMA Controller via the DMA Fast Message Read register set. Note the DMA Controller must read all 3 DMA Read Registers to obtain the complete message, unlike the case when FIFO is enabled and message is read out from the DMA Fast Message Data Read register only. The data flow will happen as follows:

- A DMA request will be asserted when any of the DMA enabled channels has successfully received one complete Fast Message. The request is kept asserted till all messages (across DMA enabled channels) have been read out.
- DMA Controller should read the complete message by accessing each register of the DMA Fast Message Read register set, the registers being:
 - [DMA Fast Message Data Read Register \(SRX_DMA_FMSG_DATA\)](#)
 - [DMA Fast Message CRC Read Register \(SRX_DMA_FMSG_CRC\)](#)
 - [DMA Fast Message Time Stamp Read Register \(SRX_DMA_FMSG_TS\)](#)
- A fixed round robin sequencing runs, as shown in [Figure 48-82](#), and this logic checks for messages in sequence 0→1→3→5→7 and back to channel 0 and so on. Channel 2 is skipped as it is not DMA enabled) and channel 4 and 6 are skipped as message is not received completely. This sequence continues like this.
- The sequence in which messages are read through DMA interface is not deterministic and depends on the sequence in which they are completely received.
- When the sequencing logic detects a message ready to be read, it will update the DMA Fast Message Read register set for DMA read and move to the next channel when this message is read out completely (i.e., 3 read accesses to the above mentioned registers)
- The DMA controller should be programmed to read only one message in one DMA transfer (i.e. 12 bytes). If it is programmed to read more than one message in one DMA transfer, underflow in buffers might occur
- User software should ensure there is no overflow in channels. In case overflow occurs, the message in the channel buffer will get overwritten by the newly received message. However, when a DMA read is in progress, the buffers will not be overwritten if overflow occurs to maintain data integrity.
- The Round Robin sequencing logic loops through each DMA enabled channel and halts at any channel having data till it is read out by the DMA Controller.

Functional description

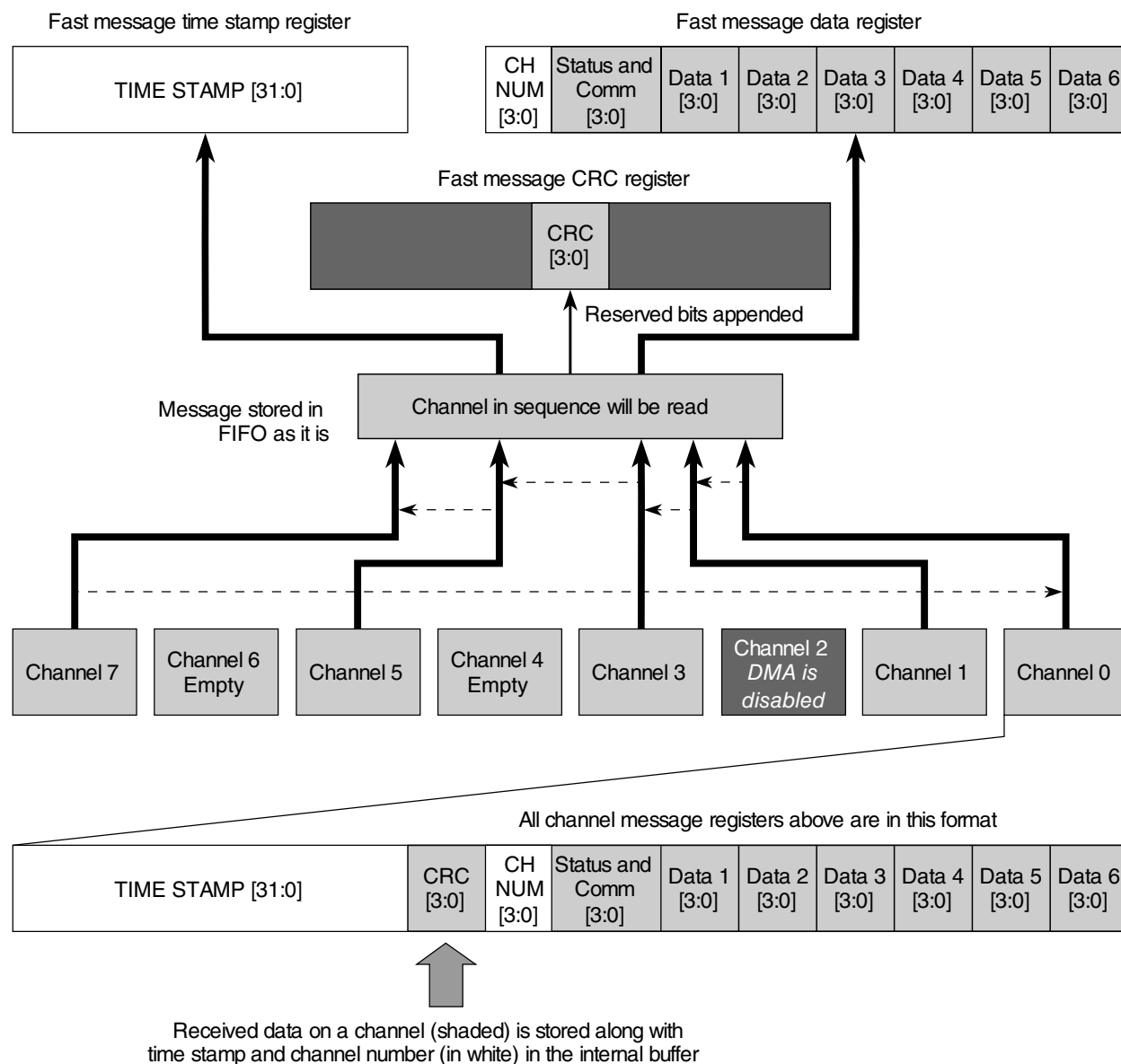


Figure 48-82. Fast Message DMA read logic (when FIFO is disabled)

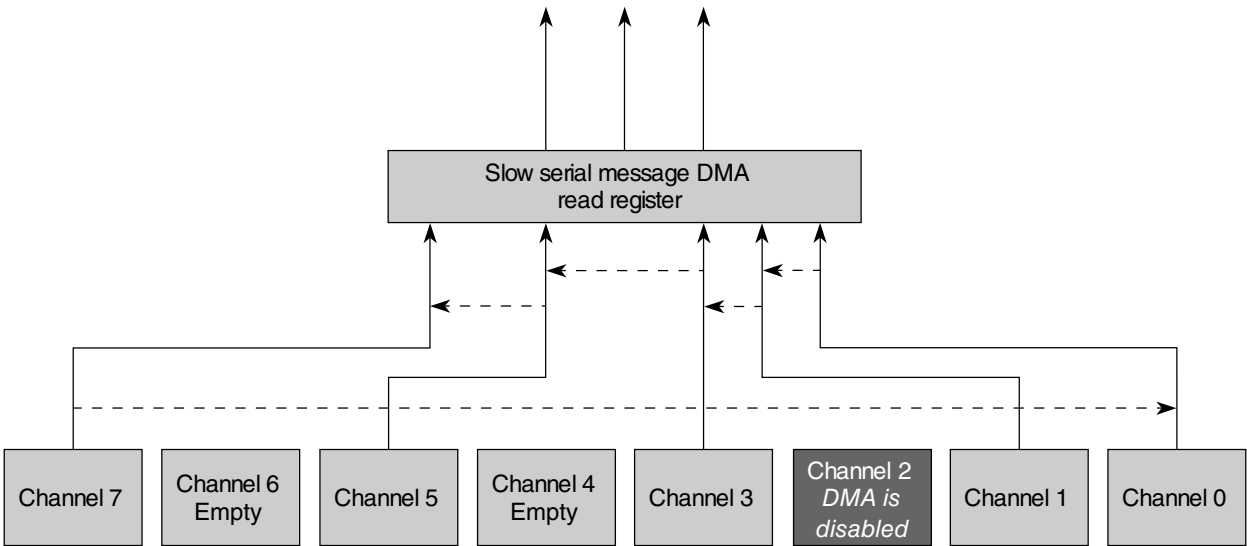
48.5.2.3 DMA request for Slow Serial Message reading

A separate DMA request (different from DMA request for Fast Message) is generated for all slow serial messages received on all channels. The data flow will happen as follows:

- Request will be asserted when any of the DMA enabled channels has successfully received one complete serial message (short or enhanced) and kept asserted till all messages have been read from their respective memory-mapped registers

- Similar to DMA read for Fast Messages, data for slow serial messages will be read from DMA Slow Serial Message Read register set in memory map, the register set comprising of:
 - [DMA Slow Serial Message Bit3 Read Register \(SRX_DMA_SMSG_BIT3\)](#)
 - [DMA Slow Serial Message Bit2 Read Register \(SRX_DMA_SMSG_BIT2\)](#)
 - [DMA Slow Serial Message Time Stamp Read Register \(SRX_DMA_SMSG_TS\)](#)
- A fixed round robin sequencing logic will be running and it will update the DMA Slow Serial Message Read register set for DMA read and move to the next channel when this message is read out completely (i.e., 3 DMA read accesses to the above mentioned registers)
- The channel number and time stamp will also be appended for every message
- A message type bit (TYPE) will also be appended to distinguish between short and enhanced serial messages
- The DMA controller should be programmed to read only one message in one DMA transfer (i.e. 12 bytes). If it is programmed to read more than one message in one DMA transfer, underflow in buffers might occur
- Since serial messages receive rate is very slow, user software should ensure there is no overflow
- The fixed round robin sequence logic for DMA read is the same as that used for Fast Messages (see [DMA request for Fast Message reading when FIFO is disabled](#))

[Figure 48-83](#) shows the sequencing of channels for serial messages to be read when DMA read is done on DMA Slow Serial Message Read register set.



Note: The arrows in the figure indicate next DMA transfer. One DMA Read Register set (3 registers) will be read in

Figure 48-83. Slow Serial Message DMA read logic

The formats in which the 3 types of Serial Messages will be stored in the buffers are shown in figures below.

Table 48-83. Enhanced Serial Message with 8-bit ID field

31		1		16				15		0		0			
CH NUM [3:0]		P							0	C	ID [7:4]		0	ID [3:0]	0
				CRC [5:0]						Data Field [11:0]					
TIME STAMP [31:0]															

Table 48-84. Enhanced Serial Message with 4-bit ID field

31		1	16				15		1		0			
CH NUM [3:0]		P							0	C	ID [3:0]	0	Data Field [15:12]	0
			CRC [5:0]						Data Field [11:0]					
TIME STAMP [31:0]														

Table 48-83 and Table 48-84 shows the enhanced serial messages. Bit C (shown in figure) or CFG (as shown in DMA Slow Serial Message Bit3 Read Register (DMA_SMSG_BIT3) description) defines the type of enhanced packet and bit P (shown in figure) or bit TYPE (shown in DMA Slow Serial Message Bit3 Read Register (DMA_SMSG_BIT3) description) defines whether it is short serial or enhanced serial message. CRC, ID and Data fields are stored in format as received and shown in SAE specification with just one exception that CRC is shifted to align with 16-bit boundary.

Table 48-85 shows the short serial message. Field locations have been placed in such a way to allow a common register to read all the slow serial messages and also match the bit positions in actual message defined by SAE Specification.

Table 48-85. Short serial message

31	0	16	15	0
CH NUM [3:0]	P			
		CRC [3:0]	ID [3:0]	Data Field [7:0]
TIME STAMP [31:0]				

Again, three 32-bit accesses to be made by DMA to get one complete message with time stamp.

48.5.3 Message reading via interrupts

There is one interrupt request per channel for a Fast message received and one for a serial message received. Each channel is individually configurable to enable or disable interrupt on reception of Fast or Slow Serial messages separately. In case of interrupt request of Fast Message, an interrupt will be asserted when any of the channels has received a complete Fast Message without errors and that channel is not configured for DMA Reads. The interrupt routine should be coded in such a way to take the least amount of overhead (in CPU cycles) while entering the routine and start reading the available messages. The routine should be capable of accessing all channels that are enabled to give out messages via interrupts through the Register Interface.

In response to these interrupts, the CPU is expected to read interrupt status register for fast or Slow Messages (depending on which interrupt was asserted) and store it in system memory. Using this status register value, the interrupt routine can decide from which channels the message is to be read. For instance if bit corresponding to channel 0 is set then routine should read from the address location corresponding to Channel 0 Fast Message Read registers and move onto the next asserted bit. The interrupt routine should scan through all the bits of this stored interrupt status register value and read corresponding channel's fast or Slow Message registers. In this manner all available messages can be read. After scanning through all the bits of status register stored in system memory, routine can read again the message ready interrupt status register to find out if any other channel has received in the mean time. This will save another interrupt routine time mainly from stacking/unstacking delay.

It is assumed that the interrupt controller will be able to respond fast enough to avoid any overflow as only one message per channel will be stored and the system bus clock frequency is sufficient to avoid this.

48.5.3.1 Suggested software interrupt service routine steps

These are suggested step to have an interrupt routine that can read all messages quickly and avoid any underflow/overflow. Actual implementations can vary.

When separate Fast Message and Slow Serial Message Interrupts are used:

The Fast Message Interrupt should have higher priority than Slow Serial Message Interrupt. The suggested ISR steps for a Fast Message Interrupts can be

- Read the Fast Message Ready Register and store it in a variable
- Shift out bits starting from MSB from this variable
- If bit is 1, perform read operation on corresponding channel's memory-mapped Fast Message Read Register set (3 registers). Three read accesses will fetch complete message. Shift out new bit. Repeat Step.
- Else if bit is 0, shift out new bit
- When all bits are shifted out, read the Fast Message Status Registers again to ensure no new message has arrived during the ISR.
- If there are no bits set, exit the ISR, else repeat this process.

The ISR steps for Slow Serial Message Interrupt can be same as for Fast Message ISR.

When common Fast Message and Slow Serial Message Interrupts are used:

The suggested ISR steps can be

- Read the Fast Message Status Register and store it in a variable
- Shift out bits starting from MSB from this variable
- If bit is 1, perform read operation on corresponding channel's memory-mapped Fast Message Read Register. Three read accesses will fetch complete message. Shift out new bit. Repeat Step.
- Else if bit is 0, shift out new bit
- When all bits are shifted out, read the Slow Serial Message Status Register into a new variable

- Shift out bits starting from MSB from this new variable
- If bit is 1, perform read operation on corresponding channel's memory-mapped Slow Serial Message Read Register. Three read accesses will fetch complete message. Shift out new bit. Repeat Step.
- Else if bit is 0, shift out new bit
- When all bits are shifted out, software can again check both Fast and Slow Serial Message Ready register to ready any message that might have been received while the routine was being serviced, or can choose to exit the routine.

48.5.4 Overflow behavior

This section describes the overflow behavior.

48.5.4.1 Fast messages buffers/FIFO

Overflow in Fast Message buffers or FIFO can occur when the CPU or DMA does not read messages from that channel for a long period of time. The receiver channel stores two messages in its buffers and indicates an overflow when these buffers are full. When FIFO is used, overflow is indicated when the FIFO gets full and another fast message is to be stored.

When an overflow event occurs, the newly received message overwrite the previously stored message and the overflow status bit is set. However, if overflow asserts while CPU/DMA is reading that channel (i.e. all 3 read registers have not yet been read), then that message is not overwritten. While messages are being overwritten, the slow message reception is not hampered.

48.5.4.2 Slow message buffers

Overflow in Slow Message buffers can occur when the CPU or DMA does not read messages from that channel for a long period of time. The receiver channel stores two messages in its buffers and indicates an overflow when these buffers are full.

When an overflow event occurs, further reception of slow messages is halted until the buffers are read out by CPU or DMA. Since it is less likely that CPU or DMA do not read the read registers for 32 or 36 fast message length duration, an overflow in slow messages would not be as frequent as in fast messages.

48.5.5 Adjustment for variation in sensor (Tx) clock

The clock used by the Channel Receiver (referred to as Receiver Clock, or Rx_CLK) to sample the message nibbles is generated from a High Frequency Receiver Clock or Protocol Clock. This high frequency receiver clock is divided using a prescaler counter which generates a tick when it rollovers on reaching a count equal to the Prescaler Value programmed in the channel's clock control register. The receive clock is generated separately for each channel. By varying this Prescaler Factor, we can control the period of the receive clock and make it close to the Transmitter clock used by the sensor for the particular channel.

After reset and once channel is enabled, the compensated prescaler value is set to Zero and the user software programs the Prescaler Value to obtain the desired receiver clock frequency for the particular channel. Thus, the receiver starts working by assuming that the transmitter is sending messages on the frequency that is programmed in its channel clock control register. The channel's input after filtering is continuously measured by the Prescaler Counter. The compensation logic checks each pulse length and determines the calibration pulse from them. When a calibration pulse is detected, the compensated prescaler value is computed and stored in the Channel's Clock Control Register ([Channel 'n' Clock Control Register \(n = 0 to \(CH-1\)\) \(SRX_CHn_CLK_CTRL\)](#)). Software can use this value to determine the variation in Tx Clock.

Note

A pause pulse may incidentally be detected as a synchronization/calibration pulse during start up after reset. The correction value would be computed on the pause pulse too but since the next synchronization pulse would follow, the logic would again detect the falling edge on the synchronization pulse and the correction value would now be correctly computed on the correct synchronization pulse.

The Clock Compensation or Correction happens for every messages that is received and the compensated prescaler value remains constant for that message.

48.5.5.1 Adjustment for nibble length variation

Section 6.2 of SAE Specifications specifies the maximum allowable limits for clock drift and jitter for the transmitter and receiver separately. This can cause the length of nibbles to vary and can cause incorrect sampling of data nibbles. This variation is adjusted by the SENT Receiver when nibbles are being sampled.

48.5.6 Input programmable filter

The Input Programmable Filter ensures that only valid input pin transitions are received by the SENT Receiver module. The input programmable filter is an 8-bit programmable up counter that increments on the high frequency receiver clock. The sensor input signal from device's pad is synchronized by high frequency receiver clock. When a state change occurs in the sensor input signal, the 8-bit counter resets and starts counting up on the high frequency receiver clock. As long as the new state is stable on the pin, the counter remains incrementing. If a counter overflow occurs, the new pin value (i.e. sensor input) is validated and latched in the filtered output flop (also running on high frequency receiver clock). If the opposite edge appears on the sensor input before validation (i.e. overflow), the counter is reset and the input is not passed onto the filtered output. At the next pin transition, the counter starts counting again. Any pulse that is shorter than a full range of the masked counter is regarded as a glitch and it is not passed on to the filter output. The number of samples (or the match value of the up counter) is programmable by the user in the channel's configuration register [Channel 'n' Configuration Register \(n=0 to \(CH-1\)\) \(SRX_CHn_CONFIG\)](#). When zero is programmed into the registers, it bypasses the filter and the synchronized input is output as the filtered output. A non-zero value delays the filtered output by the number of clocks proportional to the programmed value in the Channel Configuration Register ([Channel 'n' Configuration Register \(n=0 to \(CH-1\)\) \(SRX_CHn_CONFIG\)](#)). A timing diagram of the input filter when FIL_CNT is 4, is shown in the following figure.

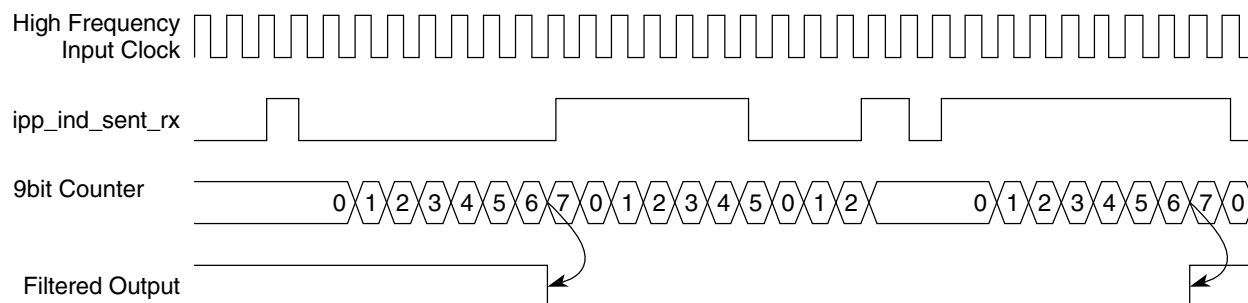


Figure 48-84. Input programmable filter timing diagram

48.5.7 Receiver diagnostics

Most of the receiver diagnostics are controlled by the Fast Message State Machine. The following checks are conducted by the receiver in each channel

- Calibration pulse length < 56 clock ticks – 25% or > 56 clock ticks + 25%

- Not the expected number of falling edges between calibration pulses. (Message length is pre-defined by each sensor device and programmed by user for each channel)
- Checksum error. Two 4-bit CRC checks and one 6-bit CRC check according to the message type. User has a programmable option to select the method of CRC to use i.e. Legacy or Recommended.
- Any nibble data values measured as < 0 or > 15
- Successive calibration pulses differ by $> +1.5625\%$ (1/64) or $< -1.5625\%$. The accuracy of this check will depend on the frequency of the protocol clock used.
- For messages with pause pulse, an additional diagnostic is done; however, this diagnostic will not fail any message (to reduce latency in message reception) but the failure of this diagnostic will be indicated in the channel's status register.

On detection of any of the above errors, the receiver rejects the current packet until a new calibration pulse is detected. Occurrence of an error is flagged in the respective channel's status register ([Channel 'n' Status Register \(n=0 to \(CH-1\)\) \(SRX_CHn_STATUS\)](#)). If error interrupt is enabled, an error interrupt will be generated. There is a single error interrupt generated for all errors, so the CPU must read the status register to find out which error occurred. Once error condition is flagged, subsequent errors (except NUM_EDGES_ERR) are automatically masked out until a valid calibration pulse is detected, after which all checks are automatically enabled. This is done to avoid multiple assertions of the interrupt to CPU in case the channel become unstable and send wrong data continuously.

48.5.7.1 Calibration pulse length check

The calibration pulse length check happens in conjunction with Clock Compensation Logic since current calibration pulse is detected in this block. The Clock Compensation Logic detects a calibration pulse of length between 42 and 70 tick counts. This check will assert the CAL_LEN_ERR status bit on failure.

48.5.7.2 Successive calibration pulse check

The successive calibration pulse check checks if successive calibration pulses differ by $> +1.5625\%$ (1/64) or $< -1.5625\%$. The accuracy of this check will depend on the frequency of the protocol clock used. Specifically, the higher the frequency, the closer the check will be to 1.5625%. So if protocol clock is T_{HF_CLK} and channel tick period is

T_{TX_CLK} then the check will be of $1.5625\% \pm \{100 \times (4 \times T_{HF\_CLK}) / (42 \times T_{TX\_CLK})\}$. So it won't allow variance in clock of a channel by $1.5625\% - \{100 \times (4 \times T_{HF\_CLK}) / (42 \times T_{TX\_CLK})\}$. This check will assert the CAL_DIAG_ERR status bit on failure.

When Option 2 of successive calibration pulse check is enabled, the CAL_RESYNC bit will be asserted on every third successive error detected by this check.

48.5.7.3 Not the expected number of edges check

The number of negative edges between two successive calibration pulses is counted and compared with expected number of edges in case of preferred option of successive calibration pulse check. The expected number of nibbles is the sum of following:

- Status and Communication nibble (1 No.)
- Data nibbles (Programmable as per user software)
- CRC nibble (1 No.)
- Pause Pulse, if configured (1 No.)

This check ("Not the expected number of edges check") will assert the NUM_EDGES_ERR status bit on failure. The check does not run when Option 2 (low latency) of successive calibration pulse check is enabled. This check is not masked after any other error is detected so it could be redundant if it is flagged along with CAL_DIAG_ERR and CAL_LEN_ERR. However, this is true only when CPU is clearing error status bits as and when they are asserted. If errors are accumulated, the above cannot be deduced.

48.5.7.4 Nibble value check

As part of this check, the lengths of the following nibbles are checked to remain in between 0 and 15 nibble values(11.5 ticks to 27.5 ticks):

- Status and Communication nibble
- all Data nibbles
- CRC nibble

The nibble value check will assert the NIB_VAL_ERR status bit on failure. This error can be flagged along with NUM_EDGES_ERR in which case the former can be ignored as the number of pulses in the message is incorrect. However, this is true only when CPU is clearing error status bits as and when they are asserted. In case, errors are accumulated, the above cannot be deduced.

48.5.7.5 CRC check

The CRC check will compute both 4-bit and 6-bit CRC as requested by State Machine control logic. Two 4-bit CRCs and one 6-bit CRC are supported to compute the CRC on Fast/Slow Serial Message (4-bit CRC) and Enhanced Serial Message (6-bit CRC) respectively. The SENT Receiver Module allows the user to calculate the 4-bit CRC using either the legacy CRC implementation method or the new recommended CRC implementation via a control bit in the channel's corresponding control register. Please refer to SAE SENT Specification for details on CRC algorithm.

Asserting the CHn_CONFIG_REG[11] bit, causes the Status and Communication nibble to be included in the 4-bit CRC calculation for Fast Serial Messages. De-asserting this bit enables normal operation as described above.

This check will assert the FMSG_CRC_ERR or SMSG_CRC_ERR status bit on failure, depending on whether the check is run for Fast Message or Slow Message.

The FMSG_CRC_ERR can be flagged along with the NUM_EDGES_ERR, in which case the CRC error can be ignored as the number of pulses in the message was incorrect. However, this is true only when CPU is clearing error status bits as and when they are asserted. If errors are accumulated, the above cannot be deduced.

48.5.7.6 Pause pulse diagnostic

The pause pulse diagnostic checks whether the ratio of calibration pulse length to full message length (including pause pulse) varies from one message to another by less than 1.5625% (as per spec). In order to have reduced latency of operation, message reception is not gated due to this check and no messages are discarded. However, when this check fails, a status bit is asserted to let the software know of this condition. The successive calibration pulse check can be configured to operate along with this check. This configurability is added to avoid conflicting results from the two checks.

There can be some inaccuracy in measuring pause pulse length as SAE specification specifies maximum clock drift/jitter up to maximum nibble length only and the frequency of high frequency receiver clock (protocol clock) used, allows clock compensation logic

to ensure accuracy up to maximum nibble length. Hence there could be some inaccuracy in this check when sampling pause pulses that are larger than the maximum nibble pulse length.

This check will assert the PP_DIAG_ERR status bit on failure. This check does not run when reception of pause pulse is disabled.

When the Single Edge Nibble Transmission (SENT) Receiver (SRX) is configured to receive a pause pulse (Channel 'n' Configuration Register - CHn_CONFIG[PAUSE_EN] = 1) and the length of the pause pulse from the sensor is in range of the valid calibration pulse length, the NUM_EDGES error can get asserted spuriously (Channel 'n' Status Register - CHn_STATUS(NUM_EDGES_ERR) = 1) when there is any diagnostic error (other than number of expected edges error) or overflow in the incoming messages from the sensor.

Software can distinguish a spurious NUM_EDGES_ERR error from a real one by monitoring other error bits. The following tables will help distinguish between a false and real assertion of NUM_EDGES_ERR error and other errors. Software should handle the first error detected as per application needs and other bits can be evaluated based on these tables. Table 1 contains information due to erratum behavior and Table 2 below contains clarification of normal NUM_EDGES_ERR behavior.

Table 48-86. Behavior of NUM_EDGES_ERR

First Error Detected	Other error bits asserted	Cause for extra error bits getting asserted	Action
NIB_VAL_ERR	NUM_EDGES_ERR asserted twice	Upon detection of the first error, the state machine goes into a state where it waits for a calibration pulse, the first NUM_EDGES_ERR error is for the current message as the state machine does not detect an end of message. The second error comes when both the Pause pulse and the Calibration pulse are seen as back to back calibration pulses and no edges in between.	Ignore both NUM_EDGES_ERR error

Table continues on the next page...

**Table 48-86. Behavior of NUM_EDGES_ERR
(continued)**

FMSG_CRC_ERR	NUM_EDGES_ERR asserted twice	Upon detection of the first error, the state machine goes into a state where it waits for a calibration pulse, the first NUM_EDGES_ERR error is for the current message as the state machine does not detect an end of message. The second error comes when both the Pause pulse and the Calibration pulse are seen as back to back calibration pulses and no edges in between.	Ignore both NUM_EDGES_ERR errors
CAL_LEN_ERR	NUM_EDGES_ERR asserted once	Since the calibration pulse is not detected as a valid calibration pulse, the internal edges counter does not detect the end of one message and start of bad message (which has CAL_LEN_ERR); hence the NUM_EDGES_ERR gets asserted.	Ignore NUM_EDGES_ERR error

Table 48-87. Expected behavior, clarification of NUM_EDGES_ERR cases

First Error Detected	Other error bits asserted	Cause for extra error bits getting asserted	Action
NUM_EDGES_ERR (when edges are less than expected)	NIB_VAL_ERR is asserted	When the actual number of edges in the message are less than expected, then a pause pulse gets detected as a nibble since the state machine expects nibbles when actually there is a pause pulse present. This generates NIB_VAL_ERR.	Ignore the NIB_VAL_ERR
NUM_EDGES_ERR (when edges are more than expected)	NIB_VAL_ERR and PP_DIAG_ERR are asserted	When the actual number of edges in a message are more than expected, then after receiving the programmed number of data nibbles, the state machine expects a pause pulse. However, the pause pulse comes later and gets detected as a nibble and hence NIB_VAL_ERR is asserted. Since the message length is not correct, PP_DIAG_ERR is also asserted.	Ignore NIB_VAL_ERR and PP_DIAG_ERR

48.5.8 Time stamp logic

A single counter maintains the time stamp for all channels which can be used by the software to determine the relative time of arrival of the messages for each channel. The time stamp counter is a 32-bit counter.

The clock for this counter is derived from the high frequency receiver clock using a Time Stamp Prescaler Counter that divides the high frequency receiver clock to generate the clock of required resolution. The user must program the time stamp prescaler value (in [Global Control Register \(SRX_GBL_CTRL\)](#)) in order to generate the time stamp clock. The period of the time stamp clock must be less than that of the fastest receiver channel being used, or it can be set at some fixed value determined by application, such as 1 μ s. For example, if the high frequency receiver clock is 60 MHz, the required prescaler value is 59 to generate a time stamp clock of 1 μ s resolution.

Note

The rollover of time stamp values should be determined in user software, by comparing the time stamp values in previous and current messages that are read out. User software should adjust the time stamp values in these messages accordingly, after reading them.

The time stamp for a Fast Message is sampled at the falling edge that comes at the end of the calibration pulse (which is also the start of the status and communication nibble pulse) and is stored in the message read buffer. However, if the current message is found to be in error, the message buffer is discarded. Thus, the time stamp will be available to user software for only those messages which do not have any error.

The time stamp for a serial message is sampled at the end of the message i.e. when the serial message is completely received and when the message is found to have no errors. the time stamp is stored in the message read buffer to be read by the user software. Again, for messages with error the message and time stamp are discarded.

48.5.8.1 Limitation

Since the same counter is used for all channels, it may happen that more than 1 message is received within the same time stamp across different channels. Hence the software would not be able to determine the sequence of arrival of messages which have the same time stamp value.

48.5.9 Bus Idle Diagnostic

This diagnostic checks the connected sensors for inactivity. SENT module can indicate if the idle period on a particular channel has crossed the period defined by the programmed value in Bus Idle Count (in [Channel 'n' Configuration Register \(n=0 to \(CH-1\)\) \(SRX_CH_n_CONFIG\)](#)). Status bit (bus_idle in [Channel 'n' Status Register \(n=0 to \(CH-1\)\) \(SRX_CH_n_STATUS\)](#)) will be set when idle period crosses the allowed value. CPU should write 1 to this clear this bit once it is read.

48.6 Clocks and resets

The SENT Receiver works on two clocks. First is the System Bus Clock that is used to program the registers and read messages via DMA or Interrupt from the module's Register Interface. Second is the High Frequency Receiver Clock or Protocol Clock that is used for accurate message receiving operations. There is a single asynchronous reset which is the system asynchronous reset and one internal reset for each of the channel logic.

48.6.1 Clocking strategy

The following figure shows the clock domain in which each module is functioning.

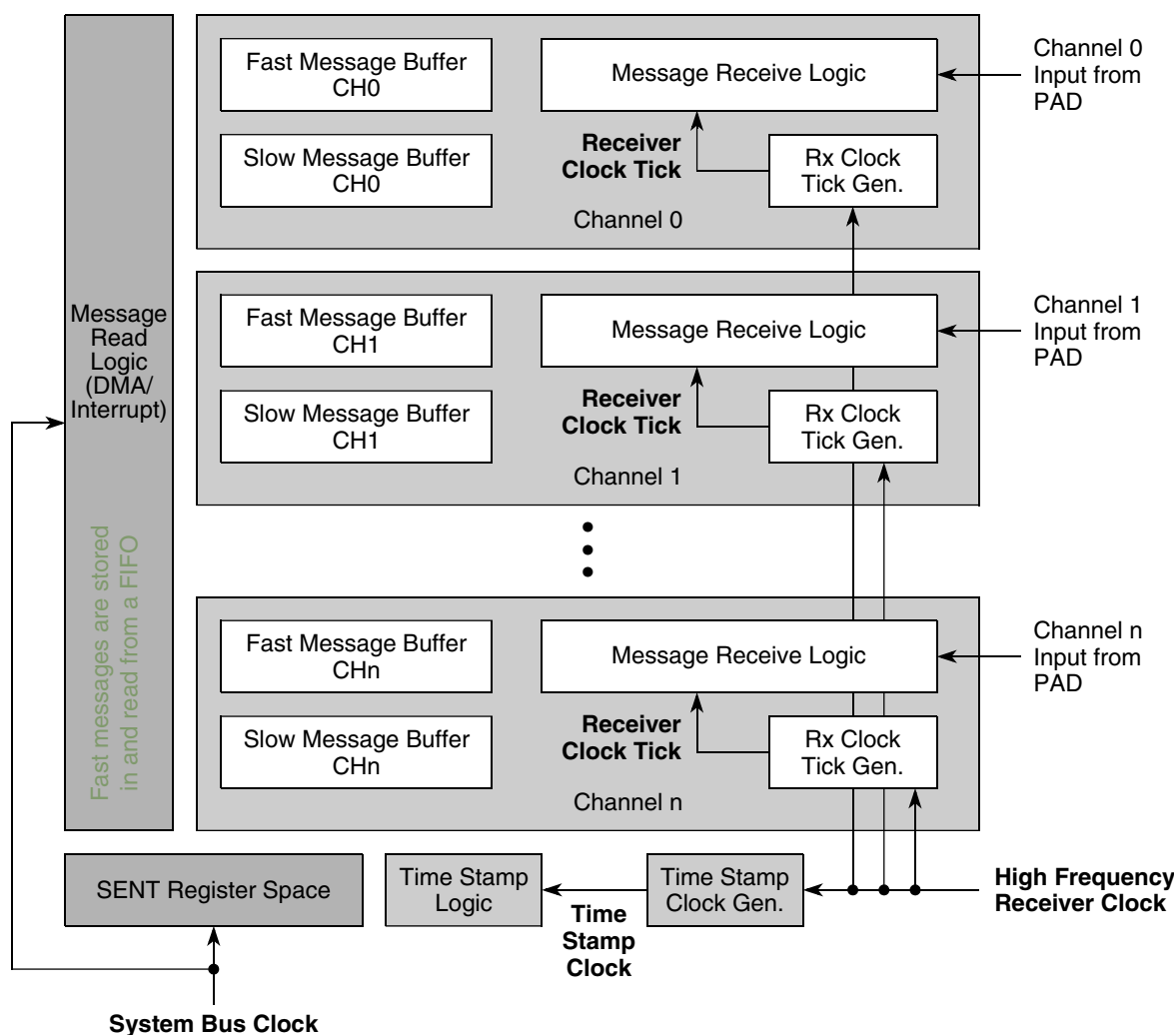


Figure 48-85. SENT Receiver module clock domains

48.6.2 System bus clock requirements

Since all the received messages are read on the System Bus Clock it is necessary that both the interrupt service routine and the DMA controller must be able to access all channels with messages ready to be read within a time to prevent overflow in any channel. The minimum time available to the CPU or DMA is the case when the smallest Fast Message is received back to back on all channels while working on the smallest receive clock tick period.

Now, the smallest Fast Message size with valid checksum as per SAE Specifications is 154 ticks. At 3 μ s receive clock tick with a -20% variation in the transmitter clock tick, the effective receiver clock tick is 2.4 μ s (after compensation). This gives us the time in which the minimum size Fast Message is received in any channel as $154 \times 2.4 = 369.6$

μs. Now since messages are arriving in parallel, 369.6 μs is the time available for CPU or DMA to access all channels to prevent an overflow in the channel that received the very first message.

Thus, if assume that ' N_{CHNL} ' channels are being used and all channels have messages ready to be read together from either DMA or CPU then the time available to read a message in one channel will be $369.6/N_{CHNL}$ μs or $\sim 370/N_{CHNL}$ μs. Since the CPU or the DMA needs to make three 32-bit accesses in order to read a complete message and taking the time required for one 32-bit read access as ' T_{RD} ' (assuming it to be same for DMA and CPU); we can determine the minimum frequency for the System Bus Clock as:

For CPU,

$$3 \times T_{RD} + T_{ISR} + T_{WAKUP} < 370 / N_{CHNL}$$

where T_{ISR} is the overhead in initiating an ISR and T_{WAKUP} is the time to wakeup from low power mode

For DMA,

$$3 \times T_{RD} + T_{WAKUP} < 370 / N_{CHNL}$$

Replacing $T_{RD} = N_{RD} \times T_{SYS\_CLK}$ and $T_{ISR} = N_{ISR} \times T_{SYS\_CLK}$, where N_{RD} is the number of System Bus Clock cycles in one 32-bit read access, N_{ISR} is the number of System Bus Clock Cycles in initiating an interrupt service routine and T_{SYS_CLK} is the period of the same clock, we get

For CPU,

$$(3 \times N_{RD} + N_{ISR}) \times T_{SYS\_CLK} + T_{WAKUP} < 370 / N_{CHNL}$$

(all units is μs)

For DMA,

$$3 \times N_{RD} \times T_{SYS\_CLK} + T_{WAKUP} < 370 / N_{CHNL}$$

(all units is μs)

48.6.3 High frequency receiver clock (protocol clock) requirements

For accurate receive operations, the protocol is required to satisfy a minimum frequency defined as per formula: Guard Band > $2.5 \times T_{HF\_CLK}$

where T_{HF_CLK} is the High Frequency Receiver Clock or Protocol Clock in ns and Guard Band is guard band in ns as specified in SAE specification, at particular transmitter uTick.

Note

T_{HF_CLK} in the above equation should be chosen based on the fastest sensor clock (with minimum guard band as specified in SAE spec) among all channels in the device.

Chapter 49

Enhanced Serial Communication Interface (eSCI)

49.1 Chip-specific eSCI information

This chip has six instances of the eSCI module. Some feature details vary among the instances.

The following table summarizes the feature differences. The table does not list feature details that the instances share.

Table 49-1. eSCI instance feature differences

Instance	DMA support
eSCI_A and eSCI_B	Yes
eSCI_C, eSCI_D, eSCI_E, and eSCI_F	No: descriptions of eSCI DMA functionality do not apply to these instances

NOTE

For eSCI_D, the single wire feature does not apply for TX/RX via PCSA3 because this pad works only as an output.

49.1.1 eSCI clocking

For information about eSCI clock sources and the eSCI's module clock (MCLK), see the [Clock generation](#) diagram, the [Peripheral clocks](#) diagram, and other details in the Clocking chapter.

49.1.2 eSCI registers: core access in supervisor mode

On this chip, all core accesses to eSCI registers occur in supervisor mode.

49.2 Introduction

The eSCI block is an enhanced SCI block with a LIN master interface layer and DMA support. The LIN master layer complies with the specifications LIN 1.3, LIN 2.0, LIN 2.1, and SAE J2602/1.

49.2.1 Bibliography

- LIN Specification Package Revision 1.3; December 12, 2002
- LIN Specification Package Revision 2.0; September 23, 2003
- LIN Network for Vehicle Applications, SAE J2602/1, September 1, 2005
- LIN Specification Package Revision 2.1; November 24, 2006

49.2.2 Acronyms and Abbreviations

The following table contains acronyms and abbreviations used in this document.

Table 49-2. Acronyms and Abbreviations

Term	Description
eSCI	Enhanced SCI block with LIN support and DMA support
SCI	Serial Communications Interface
LIN	Local Interconnect Network - A protocol for low-cost automobile networks
LIN PE	LIN Protocol Engine, Finite State Machine to control logic of the LIN hardware.
MCLK	Module Clock, defined in Module Clock
TCLK	Transmitter Clock, defined in Transmitter Clock
RCLK	Receiver Clock, defined in Receiver Clock
RSC	Receiver Sample Counter, defined in Receiver Clock

49.2.3 Glossary

Table 49-3. Glossary

Term	Definition
Logic level one	The voltage that corresponds to Boolean true (1) state.

Table continues on the next page...

Table 49-3. Glossary (continued)

Term	Definition
Logic level zero	The voltage that corresponds to Boolean false (0) state.
Set	To set a bit or bits means to establish logic level one on the bit or bits.
Clear	To clear a bit or bits means to establish logic level zero on the bit or bits.
Asserted	A signal that is asserted is in its active state. An active low signal changes from logic level one to logic level zero when asserted, and an active high signal changes from logic level zero to logic level one.
Preamble	The term preamble describes an idle character which is <i>transmitted</i> by the eSCI module.
Bit time	Duration of a single bit in a transmitted byte field or character, equivalent to the duration of one transmitter clock cycle defined in Transmitter Clock
frame	Entity that consists of the start bit followed by payload bits followed by one or more stop bits
LIN byte field	Special instance of a frame
SCI frame	Special instance of a frame
LIN frame	Sequence of break character followed by LIN byte fields
LIN TX frame	A LIN frame with the frame header, data byte fields, and checksum field transmitted by the eSCI module
LIN RX frame	A LIN frame with the header field transmitted by the eSCI module and the data byte fields and checksum field received by the eSCI module
module is idle	Module is idle, described in Module Idle Condition

49.2.4 Overview

The eSCI block allows asynchronous serial communications with peripheral devices and other CPUs. It includes special support to interface to LIN slave devices.

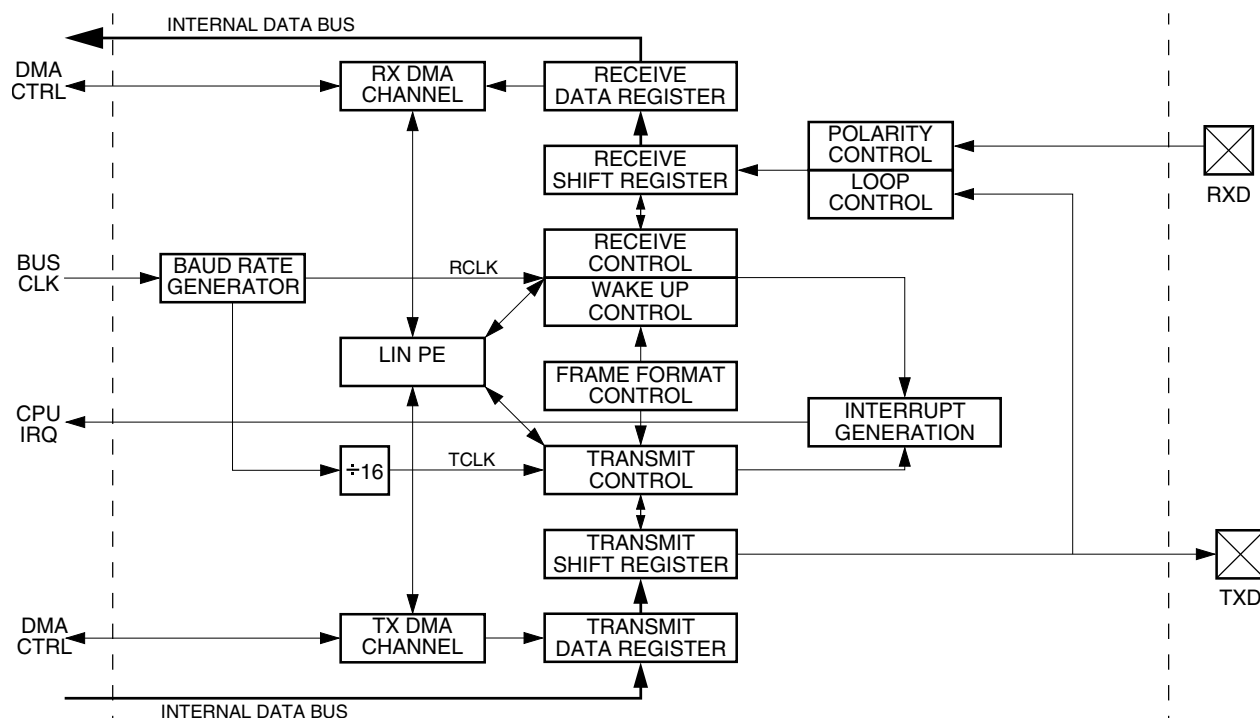


Figure 49-1. eSCI Block Diagram

49.2.5 Features

The eSCI block includes these distinctive features:

- Full-duplex operation
- Standard mark/space non-return-to-zero (NRZ) format
- 13-bit baud rate selection
- Programmable frame, payload, and character format
- Support of 2 stop bits in receiver path
- Hardware parity generation and checking
 - Programmable even or odd parity
- Programmable polarity of RXD pin
- Separately enabled transmitter and receiver
- Two receiver wake-up methods:

- Idle line wake up
- Address mark wake up
- Interrupt-driven operation with eight flags:
 - Transmitter empty
 - Transmission complete
 - Receiver full
 - Idle receiver input
 - Receiver overrun
 - Noise error
 - Framing error
 - Parity error
- Receiver framing error detection
- 1/16 bit-time noise detection
- 2 channel DMA interface
- LIN support
 - LIN Master Node functionality (master and slave task)
 - Compatible with LIN slaves from revisions 1.x and 2.0 of the LIN standard
 - Detection of Bit Errors, Physical Bus Errors and Checksum Errors
 - All status bit can generate maskable interrupts
 - Application layer CRC support
 - Programmable CRC polynomial
 - Detection and generation of wake-up characters
 - Programmable wake-up delimiter time
 - Programmable slave timeout
 - Can be configured to include header bits in checksum
 - LIN DMA interface

49.2.6 Modes of Operation

The eSCI module has two functional operational modes, SCI and LIN mode. The availability of register bits and fields depends on the selected operational mode.

49.2.6.1 Module Idle Condition

Doze mode, Disable mode and Stop mode can only be entered when the module is idle. The module is idle if

- all five active status bits DACT, WACT, LACT, TACT, and RACT in the Interrupt Flag and Status Register 1 (eSCI_IFSR1) are 0, and
- no interrupt request is pending, i.e either the interrupt flag or its related interrupt enable is 0.

To ensure that the module goes idle, the application should clear all interrupt enable bits before triggering the mode change.

49.2.6.2 SCI Mode

The SCI mode is the default functional operational mode and is described in [SCI Mode](#).

49.2.6.3 LIN Mode

The LIN mode is the second functional operational mode and is described in [LIN Mode](#).

49.2.6.4 Disabled Mode

In the Disabled mode the eSCI module indicates to the clocking system, that all module clocks can be turned off.

The eSCI module is in the Disabled Mode, if the MDIS bit in the Control Register 2 (eSCI_CR2) is set and the module is idle.

49.3 External Signal Description

The eSCI module is connected to a total of two external pins.

49.3.1 Detailed Signal Descriptions

49.3.1.1 eSCI Transmit Pin (TXD)

This pin serves as transmit data output and as the receive data input of eSCI.

49.3.1.2 eSCI Receive Pin (RXD)

This pin serves as receive data input of the eSCI.

49.4 Memory Map and Register Definition

Writes to a reserved register location do not have any effect and reads of these locations return a zero.

NOTE

All R/W registers are writable only in supervisor mode of the core performing the write.

eSCI memory map

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
0	Baud Rate Register (eSCI_BRR)	16	R/W	0004h	49.4.1/2682
2	Control Register 1 (eSCI_CR1)	16	R/W	0000h	49.4.2/2683
4	Control Register 2 (eSCI_CR2)	16	R/W	2000h	49.4.3/2685
6	SCI Data Register (eSCI_SDR)	16	R/W	0000h	49.4.4/2687
8	Interrupt Flag and Status Register 1 (eSCI_IFSR1)	16	w1c	8000h	49.4.5/2689
A	Interrupt Flag and Status Register 2 (eSCI_IFSR2)	16	w1c	4000h	49.4.6/2691
C	LIN Control Register 1 (eSCI_LCR1)	16	R/W	0000h	49.4.7/2692
E	LIN Control Register 2 (eSCI_LCR2)	16	R/W	0000h	49.4.8/2694

Table continues on the next page...

eSCI memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
10	LIN Transmit Register (eSCI_LTR)	8	R/W	00h	49.4.9/2694
14	LIN Receive Register (eSCI_LRR)	8	R	00h	49.4.10/2696
18	LIN CRC Polynomial Register (eSCI_LPR)	16	R/W	C599h	49.4.11/2697
1A	Control Register 3 (eSCI_CR3)	16	R/W	0000h	49.4.12/2697

49.4.1 Baud Rate Register (eSCI_BRR)

This register provides the control value for the serial baud rate. The baud rate and clock generation is specified in [Baud Rate and Clock Generation](#).

A byte write access to only the upper byte of this register (eSCI_BRR[0:7]) will not change the content of the register, instead, the written byte is stored internally into a shadow register. A subsequent byte write access to only the lower byte of this register (eSCI_BRR[8:15]) updates the lower byte and copies the content of the shadow register into the upper byte.

A byte write access to only the lower byte of this register (eSCI_BRR[8:15]) without a preceding byte write access to only the upper byte copies a value of all zero into the upper byte.

A 16-bit write access to this register updates both the lower and upper byte immediately and is the recommended write access type for this register.

Address: 0h base + 0h offset = 0h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Read	0			SBR												
Write																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0

eSCI_BRR field descriptions

Field	Description
0–2 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
3–15 SBR	Serial Baud Rate. This field provides the baud rate control value SBR.

49.4.2 Control Register 1 (eSCI_CR1)

This register provides bits to configure the functionality of the module, provides the interrupt enable bits for the interrupt flags provided in [Interrupt Flag and Status Register 1 \(eSCI_IFSR1\)](#) and provides the control bits for the transmitter and receiver.

Table 49-7. Receive Source Mode Selection

LOOPS	RSRC	Receiver Input Mode
0	0	Dual Wire Mode
0	1	Reserved
1	0	Loop Mode
1	1	Single Wire Mode

Address: 0h base + 2h offset = 2h

Bit	0	1	2	3	4	5	6	7
Read	LOOPS	Reserved	RSRC	M	WAKE	0	PE	PT
Write								
Reset	0	0	0	0	0	0	0	0

Bit	8	9	10	11	12	13	14	15
Read	TIE	TCIE	RIE	ILIE	TE	RE	RWU	SBK
Write								
Reset	0	0	0	0	0	0	0	0

eSCI_CR1 field descriptions

Field	Description
0 LOOPS	Loop Mode Select. This control bit together with the RSRC control bit defines the receiver source mode. The mode coding is defined in Table 49-7 and the modes are described in Receiver Input Mode Selection .
1 Reserved	This field is reserved. Application must not write 1 to this bit.
2 RSRC	Receiver Source Control. This control bit together with the LOOPS control bit defines the receiver source mode. The mode coding is defined in Table 49-7 and the modes are described in Receiver Input Mode Selection .
3 M	Frame Format Mode. This control bit together with the M2 bit of the Control Register 3 (eSCI_CR3) controls the frame format used. The supported frame formats and the related settings are defined in Frame Formats .
4 WAKE	Receiver Wake-up Condition. This control bit defines the wake-up condition for the receiver. The receiver wake-up is described in Multiprocessor Communication . 0 Idle line wake-up. 1 Address mark wake-up
5 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
6 PE	Parity Enable. This control bit enables the parity bit generation and checking. The location of the parity bits is shown in Frame Formats .

Table continues on the next page...

eSCI_CR1 field descriptions (continued)

Field	Description
	0 Parity bit generation and checking disabled. 1 Parity bit generation and checking enabled.
7 PT	Parity Type. This control bit defines whether even or odd parity has to be used. 0 Even parity (even number of ones in character clears the parity bit). 1 Odd parity (odd number of ones in character clears the parity bit).
8 TIE	Transmitter Interrupt Enable. This bit controls the eSCI_IFSR1[TRDE] interrupt request generation. 0 TDRE interrupt request generation disabled. 1 TDRE interrupt request generation enabled.
9 TCIE	Transmission Complete Interrupt Enable. This bit controls the eSCI_IFSR1[TC] interrupt request generation. 0 TC interrupt request generation disabled. 1 TC interrupt request generation enabled.
10 RIE	Receiver Full Interrupt Enable. This bit controls the eSCI_IFSR1[RDRF] interrupt request generation. 0 RDRF interrupt request generation disabled. 1 RDRF interrupt request generation enabled.
11 ILIE	Idle Line Interrupt Enable. This bit controls the eSCI_IFSR1[IDLE] interrupt request generation. 0 IDLE interrupt request generation disabled. 1 IDLE interrupt request generation enabled.
12 TE	Transmitter Enable. This control bit enables and disables the transmitter. The control features of the transmitter are described in Transmitter States and Transitions . 0 Transmitter disabled. 1 Transmitter enabled.
13 RE	Receiver Enable. This control bit enables and disables the receiver. The control features of the receiver are described in Receiver States and Transitions . 0 Receiver disabled. 1 Receiver enabled.
14 RWU	Receiver Wake-Up Mode. This bit controls and indicates the receiver wake-up mode, which is described in Multiprocessor Communication . NOTE: This bit should be set in SCI mode only. 0 Normal receiver operation. 1 Receiver is in wake-up mode.
15 SBK	Send Break Character. This bit controls the transmission of break characters, which is described in Break Character Transmission . NOTE: This bit should be set in SCI mode only. 0 No break characters will be transmitted. 1 Break characters will be transmitted.

49.4.3 Control Register 2 (eSCI_CR2)

This register provides bits to configure the functionality of the module, and interrupt enable bits for the interrupt flags provided in [Interrupt Flag and Status Register 1 \(eSCI_IFSR1\)](#) and control bits for the transmitter and receiver.

Address: 0h base + 4h offset = 4h

Bit	0	1	2	3	4	5	6	7
Read								
Write								
Reset	0	0	1	0	0	0	0	0
Bit	8	9	10	11	12	13	14	15
Read								
Write								
Reset	0	0	0	0	0	0	0	0

eSCI_CR2 field descriptions

Field	Description
0 MDIS	Module Disabled Mode. This bit controls the Module Mode of Operation, which is described in Modes of Operation . 0 Module is not in Disabled Mode. 1 Module is in Disabled Mode, if module is idle.
1 FBR	Fast Bit Error Detection. This bit controls the Bit Error Detection mode. NOTE: This bit is used in LIN mode only. 0 Standard Bit error detection performed as described in Standard Bit Error Detection . 1 Fast Bit error detection performed as described in Fast Bit Error Detection .
2 BSTP	DMA Stop on Bit Error or Physical Bus Error. This bit controls the transmit DMA requests generation in case of bit errors or physical bus errors. Bit errors are indicated by the BERR flag in the Interrupt Flag and Status Register 1 (eSCI_IFSR1) and physical bus errors are indicated by the PBERR flag in the Interrupt Flag and Status Register 2 (eSCI_IFSR2) . NOTE: This bit is used in LIN mode only. 0 Transmit DMA requests generated regardless of bit errors or physical bus errors. 1 Transmit DMA requests are not generated if eSCI_IFSR1[BERR] flag or eSCI_IFSR2[PBERR] flags are set.
3 BERRIE	Bit Error Interrupt Enable. This bit controls the BERR interrupt request generation. 0 BERR interrupt request generation disabled. 1 BERR interrupt request generation enabled.
4 RXDMA	Receive DMA Control. This bit enables the receive DMA feature. When this bit is cleared, a pending receive DMA request is deasserted.

Table continues on the next page...

eSCI_CR2 field descriptions (continued)

Field	Description
	0 Receive DMA disabled. 1 Receive DMA enabled.
5 TXDMA	Transmit DMA Control. This bit enables the transmit DMA feature. When this bit is cleared, a pending transmit DMA request is deasserted. 0 Transmit DMA disabled. 1 Transmit DMA enabled.
6 BRCL	Break Character Length. This bit is used to define the length of the break character to be transmitted. The settings are specified in Break Character Formats .
7 TXDIR	TXD pin output driver control. This bit has no effect. In this device, TXDIR does not control the output driver of the TXD pin in Single Wire Mode. The function is controlled by the relevant Pad Control Register in the System Integration Unit.
8 BESM	Fast Bit Error Detection Sample Mode. This bit defines the sample point for the Fast Bit Error Detection Mode. NOTE: This bit is used in LIN mode only. 0 Sample point is RS9. 1 Sample point is RS13.
9 BESTP	Bit Error Transmit Stop. This control bit defines the behavior of the eSCI Transmit Pin TXD while the bit error flag eSCI_IFSR1[BERR] is 1. NOTE: This bit is used in LIN mode only. 0 Application Data Values driven onto TXD pin. 1 Recessive Data Value 1 driven onto TXD pin.
10 RXPOL	RXD Pin polarity. This bit controls the polarity of the RXD pin. See Inverted Data Frame Formats . 0 Normal Polarity. 1 Inverted Polarity.
11 PMSK	Parity Bit Masking. This bit defines whether the received parity bit is presented in the related bit position in the SCI Data Register (eSCI_SDR) . 0 The received parity bit is presented in the bit position related to the parity bit. 1 The value 0 is presented in the bit position related to the parity bit.
12 ORIE	Overrun Interrupt Enable. This bit controls the eSCI_IFSR1[OR] interrupt request generation. 0 OR interrupt request generation disabled. 1 OR interrupt request generation enabled.
13 NFIE	Noise Interrupt Enable. This bit controls the eSCI_IFSR1[NF] interrupt request generation. 0 NF interrupt request generation disabled. 1 NF interrupt request generation enabled.
14 FEIE	Frame Error Interrupt Enable. This bit controls the eSCI_IFSR1[FE] interrupt request generation. 0 FE interrupt request generation disabled. 1 FE interrupt request generation enabled.
15 PFIE	Parity Error Interrupt Enable. This bit controls the eSCI_IFSR1[PF] interrupt request generation.

Table continues on the next page...

eSCI_CR2 field descriptions (continued)

Field	Description
0	PF interrupt request generation disabled.
1	PF interrupt request generation enabled.

49.4.4 SCI Data Register (eSCI_SDR)

In SCI mode this register is used to provide transmit data and retrieve received data. In LIN mode any write access to this register is ignored. In case of read accesses to this register during data transmission/reception this register provides a part of the transmitted/received data.

If the application writes to the lower byte of this register (eSCI_SDR[8:15]), the internal commit flag iCMT, which is not visible to the application, is set to indicate that the register has been updated and ready to transmit new data.

If the application reads from the lower byte of this register (eSCI_SDR[8:15]), a signal is send to the internal receiver unit to indicate that the register was read and is ready to receive new data. The read access will not change the content of any register.

Address: 0h base + 6h offset = 6h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Read	RN		ERR	0												
Write		TN														
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

eSCI_SDR field descriptions

Field	Description
0 RN	Received Most Significant Bit. The semantic of this bit depends on the frame format selected by eSCI_CR3[M2], eSCI_CR1[M], and eSCI_CR1[PE]. [M2=0,M=1,PE=0]: value of received data bit 8 or address bit. [M2=0,M=1,PE=1]: value of received parity bit if eSCI_CR2[PMSK]=0, 0 otherwise. [M2=1,M=0,PE=1]: value of received parity bit if eSCI_CR2[PMSK]=0, 0 otherwise . [M2=1,M=1,PE=1]: value of received parity bit if eSCI_CR2[PMSK]=0, 0 otherwise . It is 0 for all other frame formats.
1 TN	Transmit Most Significant Bit. The semantic of this bit depends on the frame format selected by eSCI_CR3[M2], eSCI_CR1[M], and eSCI_CR1[PE]. [M2=0,M=1,PE=0]: value to be transmitted as data bit 8 or address bit. It is not used for all other frame formats.

Table continues on the next page...

eSCI_SDR field descriptions (continued)

Field	Description
2 ERR	Receive Error Bit. This bit indicates the occurrence of the errors selected by the Control Register 3 (eSCI_CR3) during the reception of the frame presented in SCI Data Register (eSCI_SDR). In case of an overrun error for subsequent frames this bit is set too. 0 None of the selected errors occurred. 1 At least one of the selected errors occurred.
3 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
4–15 RDTD	The contents of this field vary by bit and by semantic as follows. Most-significant 4 bits (RDTD[11:8]): Received Data. The semantic of this field depends on the frame format selected by eSCI_CR3[M2] and eSCI_CR1[M]. [M2=1,M=1]: value of the received data bits 11:8. (Rx=BITx) . It is all 0 for all other frame formats . Fifthmost-significant bit (RDTD[7]): When read, this is Received Bit 7. The semantic of this bit depends on the format selected by eSCI_CR3[M2], eSCI_CR1[M], and eSCI_CR1[PE]. [M2=0,M=0,PE=0]: value of received BIT7 or ADDR BIT. [M2=0;M=0,PE=1]: value of received PARITY BIT if eSCI_CR2[PMSK]=0, 0 otherwise. For all other frame formats it is the value of received BIT7. When written, this is Transmit Bit 7.The semantic of this bit depends on the format selected by eSCI_CR3[M2], eSCI_CR1[M], and eSCI_CR1[PE]. [M2=0,M=0,PE=0]: value of transmit BIT7 or ADDR BIT. [M2=0;M=0,PE=1]: not used. PARITY BIT is generated internally before transmission. For all other frame formats it is the value of transmit BIT7. Remaining bits (RDTD[6:0]): When read, these are Received bits 6 to 0. Value of received BITx is shown in bit Rx. When written, these are Transmit bits 6 to 0. Value of bit Tx is transmitted in BITx.

49.4.5 Interrupt Flag and Status Register 1 (eSCI_IFSR1)

This register provides interrupt flags that indicate the occurrence of module events. The related interrupt enable bits are located in [Control Register 1 \(eSCI_CR1\)](#) and [Control Register 2 \(eSCI_CR2\)](#).

Address: 0h base + 8h offset = 8h

Bit	0	1	2	3	4	5	6	7
Read	TDRE	TC	RDRF	IDLE	OR	NF	FE	PF
Write	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c
Reset	1	0	0	0	0	0	0	0
Bit	8	9	10	11	12	13	14	15
Read	0		DACT	BERR	WACT	LACT	TACT	RACT
Write				w1c				
Reset	0	0	0	0	0	0	0	0

eSCI_IFSR1 field descriptions

Field	Description
0 TDRE	Transmit Data Register Empty Interrupt Flag. This interrupt flag is set when the content of the SCI Data Register (eSCI_SDR) was transferred into internal shift register. NOTE: This flag is set in SCI mode only.
1 TC	Transmit Complete Interrupt Flag. This interrupt flag is set when a frame, break or idle character transmission has been completed and no data were written into SCI Data Register (eSCI_SDR) after the last setting of the TDRE flag and the SBK bit in Control Register 1 (eSCI_CR1) is 0. NOTE: This flag is set in SCI mode only. In case of no data available for transmission, it is also set if the preamble was transmitted after the enabling of the transmitter.
2 RDRF	Receive Data Register Full Interrupt Flag. This interrupt flag is set when the payload data of a received frame was transferred into the SCI Data Register (eSCI_SDR) and the receive DMA is disabled. NOTE: This flag is set in SCI mode only.
3 IDLE	Idle Line Interrupt Flag. This interrupt flag is set when an idle character was detected and the receiver is not in the wake-up state. NOTE: This flag is set in SCI mode only.
4 OR	Overrun Interrupt Flag. This interrupt flag is set when an overrun was detected as described in Receiver Overrun . NOTE: This flag is set in SCI mode only.
5 NF	Noise Interrupt Flag. This interrupt flag is set when the receiver has detected noise during the reception of a frame, as described in Bit Sampling .
6 FE	Framing Error Interrupt Flag. This interrupt flag is set when the payload data of a received frame was transferred into the SCI Data Register (eSCI_SDR) or LIN Receive Register (eSCI_LRR) and the receiver has detected a framing error during the reception of that frame, as described in Stop Bit Verification .

Table continues on the next page...

eSCI_IFSR1 field descriptions (continued)

Field	Description
7 PF	Parity Error Interrupt Flag. This interrupt flag is set when the payload data of a received frame was transferred into the SCI Data Register (eSCI_SDR) and the receiver has detected a parity error for the character, as described in Reception Error Reporting . NOTE: This flag is set in SCI mode only.
8–9 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
10 DACT	DMA Active. The status bit is set when a transmit or receive DMA request is pending . 0 No DMA request pending. 1 DMA request pending.
11 BERR	Bit Error Interrupt Flag. This flag is set when a bit error was detected as described in Standard Bit Error Detection . Note: This flag is set in LIN mode only.
12 WACT	LIN Wake-Up Active. The status bit is set as long as the LIN wakeup engine receives a LIN wake-up signal . 0 No LIN wakeup signal reception in progress . 1 LIN wakeup signal reception in progress .
13 LACT	LIN Active. This status bit is set as long as the LIN protocol engine is about to transmit or receive LIN frames . 0 No LIN frame transmission or reception in progress . 1 LIN frame transmission or reception in progress .
14 TACT	Transmitter Active. This status bit is set as long as the transmission of a frame or special character is ongoing . 0 No transmission in progress . 1 Transmission in progress .
15 RACT	Receiver Active. This status bit is set as long as the receive is active. The set and clear conditions for the SCI mode are described in Receiver States and Transitions . The set and clear conditions for the LIN mode are described in LIN byte field reception . 0 No reception in progress . 1 Reception in progress .

49.4.6 Interrupt Flag and Status Register 2 (eSCI_IFSR2)

This register provides interrupt flags that indicate the occurrence of LIN related events. The related interrupt enable bits are located in [LIN Control Register 1 \(eSCI_LCR1\)](#) and [LIN Control Register 2 \(eSCI_LCR2\)](#). All interrupt flags in this register will be set in LIN mode only .

Address: 0h base + Ah offset = Ah

Bit	0	1	2	3	4	5	6	7
Read	RXRDY	TXRDY	LWAKE	STO	PBERR	CERR	CKERR	FRC
Write	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c
Reset	0	1	0	0	0	0	0	0
Bit	8	9	10	11	12	13	14	15
Read	0						UREQ	OVFL
Write							w1c	w1c
Reset	0	0	0	0	0	0	0	0

eSCI_IFSR2 field descriptions

Field	Description
0 RXRDY	Receive Data Ready Interrupt Flag. This interrupt flag is set when the payload data of a received frame was transferred into the LIN Receive Register (eSCI_LRR) and the receive DMA is disabled. NOTE: This bit should be cleared only after reading the data from the LIN Receive Register (eSCI_LRR)
1 TXRDY	Transmit Data Ready Interrupt Flag. This interrupt flag is set when a) the content of the LIN Transmit Register (eSCI_LTR) was processed by the LIN PE to generate frame header or frame transmit data, or b) when the module has transmitted a LIN wakeup signal frame.
2 LWAKE	LIN Wake-up Received Interrupt Flag. This interrupt flag is set when a LIN Wake-up character was received, as described in LIN Wake Up .
3 STO	Slave Timeout Interrupt Flag. This interrupt flag is set when a Slave-Not-Responding-Error is detected. A detailed description is given in Slave-Not-Responding-Error Detection .
4 PBERR	Physical Bus Error Interrupt Flag. This interrupt flag is set when the receiver input remains unchanged for at least 31 RCLK clock cycles after the start of a byte transmission, as described in LIN Error Reporting .
5 CERR	CRC Error Interrupt Flag. This interrupt flag is set when an incorrect CRC pattern was detected for a received LIN frame.
6 CKERR	Checksum Error Interrupt Flag. This interrupt flag is set when a checksum error was detected for a received LIN frame.
7 FRC	Frame Complete Interrupt Flag. This interrupt flag is set when a LIN TX frame has been completely transmitted or a LIN RX frame has been completely received.
8–13 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

Table continues on the next page...

eSCI_IFSR2 field descriptions (continued)

Field	Description
14 UREQ	Unrequested Data Received Interrupt Flag. This interrupt flag is set when unrequested activity has been detected on the LIN bus, as described in LIN Error Reporting .
15 OVFL	Overflow Interrupt Flag. This interrupt flag is set when an overflow as described in Overflow Detection was detected.

49.4.7 LIN Control Register 1 (eSCI_LCR1)

This register provides control bits to control and configure the LIN hardware. This register provides the interrupt enable bits for the interrupt flags in [Interrupt Flag and Status Register 2 \(eSCI_IFSR2\)](#).

Address: 0h base + Ch offset = Ch

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Read	LRES	0	WUD		0		PRTY	LIN	RXIE	TXIE	WUIE	STIE	PBIE	CIE	CKIE	FCIE
Write		WU														
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

eSCI_LCR1 field descriptions

Field	Description
0 LRES	LIN Protocol Engine Stop and Reset. This bit is used to stop and reset the LIN protocol engine as described in LIN Protocol Engine Stop and Reset . 0 LIN protocol engine is operational. 1 LIN protocol engine is reset and stopped.
1 WU	LIN Bus Wake-Up Trigger. This bit is used to trigger the generation of a wake-up signal frame on the LIN bus, as described in LIN Wake Up . 0 Write has no effect. 1 Write triggers the generation of a wake-up signal.
2–3 WUD	LIN Bus Wake-Up Delimiter Time. This field determines how long the LIN protocol engine waits after the end of the transmitted wake-up signal, before starting the next LIN frame transmission. 00 3 bit times. 01 7 bit times. 10 31 bit times. 11 63 bit times.
4–5 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
6 PRTY	Parity Generation Control. This bit controls the generation of the two parity bits in the LIN header.

Table continues on the next page...

eSCI_LCR1 field descriptions (continued)

Field	Description
	0 Parity bits generation disabled. 1 Parity bits generation enabled.
7 LIN	LIN Mode Control. This bit controls whether the device is in SCI or LIN Mode. 0 SCI Mode. 1 LIN Mode.
8 RXIE	Receive Data Ready Interrupt Enable. This bit controls the eSCI_IFSR2[RXRDY] interrupt request generation. 0 RXRDY interrupt request generation disabled. 1 RXRDY interrupt request generation enabled.
9 TXIE	Transmit Data Ready Interrupt Enable. This bit controls the eSCI_IFSR2[TXRDY] interrupt request generation. 0 TXRDY interrupt request generation disabled. 1 TXRDY interrupt request generation enabled.
10 WUIE	LIN Wake-up Received Interrupt Enable. This bit controls the eSCI_IFSR2[LWAKE] interrupt request generation. 0 LWAKE interrupt request generation disabled. 1 LWAKE interrupt request generation enabled.
11 STIE	Slave Timeout Flag Interrupt Enable. This bit controls the eSCI_IFSR2[STO] interrupt request generation. 0 STO interrupt request generation disabled. 1 STO interrupt request generation enabled.
12 PBIE	Physical Bus Error Interrupt Enable. This bit controls the eSCI_IFSR2[PBERR] interrupt request generation. 0 PBERR interrupt request generation disabled. 1 PBERR interrupt request generation enabled.
13 CIE	CRC Error Interrupt Enable. This bit controls the eSCI_IFSR2[CERR] interrupt request generation. 0 CERR interrupt request generation disabled. 1 CERR interrupt request generation enabled.
14 CKIE	Checksum Error Interrupt Enable. This bit controls the eSCI_IFSR2[CKERR] interrupt request generation. 0 CKERR interrupt request generation disabled. 1 CKERR interrupt request generation enabled.
15 FCIE	Frame Complete Interrupt Enable. This bit controls the eSCI_IFSR2[FRC] interrupt request generation. 0 FRC interrupt request generation disabled. 1 FRC interrupt request generation enabled.

49.4.8 LIN Control Register 2 (eSCI_LCR2)

This register provides the interrupt enable bits for the interrupt flags in [Interrupt Flag and Status Register 2 \(eSCI_IFSR2\)](#).

Address: 0h base + Eh offset = Eh

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Read	0						UQIE	OFIE	0							
Write																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

eSCI_LCR2 field descriptions

Field	Description
0–5 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
6 UQIE	Unrequested Data Received Interrupt Enable. This bit controls the eSCI_IFSR2[UREQ] interrupt request generation. 0 UREQ interrupt request generation disabled. 1 UREQ interrupt request generation enabled.
7 OFIE	Overflow Interrupt Enable. This bit controls the eSCI_IFSR2[OVFL] interrupt request generation. 0 OVFL interrupt request generation disabled. 1 OVFL interrupt request generation enabled.
8–15 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

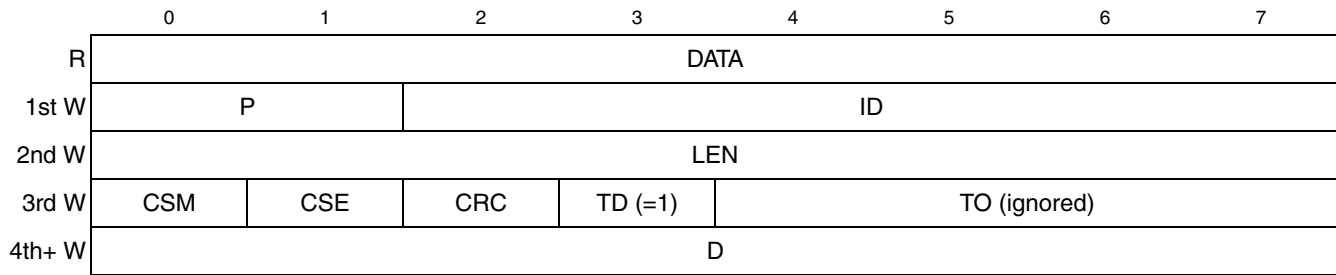
49.4.9 LIN Transmit Register (eSCI_LTR)

This register is used by the application to initiate the LIN frame header generation for both LIN TX frames and LIN RX frames. If a LIN TX frame is generated, this register is used to provide the payload data for the LIN TX frame.

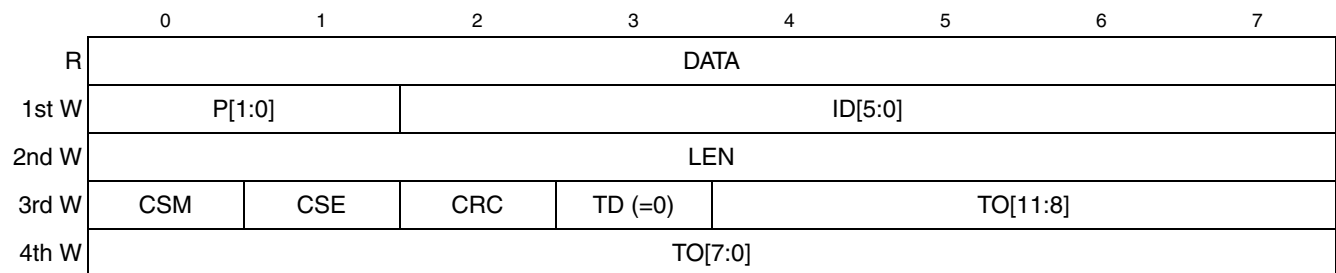
If the LIN PE is in the idle state (eSCI_LCR1[LRES] = 1) or performs a wakeup, each write access to this register is ignored.

In case of a read access, the register provides the last data written into this register in the DATA field.

If the application initiates a LIN TX frame transfer, i.e the TD bit is set to 1, the content and usage shown in [Figure 49-11](#) applies. The initiation and transmit of a TX frame is described in [LIN TX Frame generation](#).

**Figure 49-11. LIN Transmit Register (eSCI_LTR) - LIN TX frame generation**

If the application initiates an LIN RX frame, i.e the TD bit is set to 0, the content and usage shown in [Figure 49-12](#) applies. The initiation and transmit of a RX frame is described in [LIN RX frame generation](#).

**Figure 49-12. LIN Transmit Register (eSCI_LTR) - LIN RX frame generation**

Each successful write access to this register increments the internal write access counter and enables the writing to the next field. The write access counter is reset if

- the LIN PE is in the idle state (eSCI_LCR1[LRES] = 1)
- a LIN TX frame was completely transmitted (eSCI_IFSR1[FRC] was set to 1)
- a LIN RX frame was completely received (eSCI_IFSR1[FRC] was set to 1)

Table 49-15. Structure of the LTR[DATA] when written to

Field	Description
P	Identifier Parity. This field provides the identifier parity which is used to create the protected identifier if the automatic identifier parity generation is disabled, i.e the PRTY bit in LIN Control Register 1 (eSCI_LCR1) is 0.
ID	Identifier. This field is used for the identifier field in the protected identifier.
LEN	Frame Length. This field defines the number of data bytes to be transmitted or received.
CSM	Checksum Model. This bit controls the checksum calculation model used. 0: Classic Checksum Model (LIN 1.3). 1: Enhanced Checksum Model (LIN 2.0).
CSE	Checksum Enable. This bit control the generation and checking of the checksum byte. 0: No generation and checking of checksum byte. 1: Generation and checking of checksum byte.

Table continues on the next page...

Table 49-15. Structure of the LTR[DATA] when written to (continued)

Field	Description
CRC	CRC Enable. This bit controls the generation of checking standard or enhanced LIN frames, which are described in LIN frame formats . 0: Standard LIN frame generation and checking. 1: Enhanced LIN frame generation and checking.
TD	Transfer Direction. This bit controls the transfer direction of the data, CRC, and checksum byte fields. 0: Data, CRC, and Checksum byte fields received, described in LIN RX frame generation . 1: Data, CRC, and Checksum byte fields transmitted, described in LIN TX Frame generation .
TO	Timeout Value. The content of the field depends on the transfer direction. RX frame: Defines the time available for a complete RX frame transfer, as described in the note in Fast Bit Error Detection . TX frame: Must be set to 0.
D	Transmit Data . Data bits for transmission.

Address: 0h base + 10h offset = 10h

Bit	0	1	2	3	4	5	6	7
Read	DATA							
Write								
Reset	0	0	0	0	0	0	0	0

eSCI_LTR field descriptions

Field	Description
0-7 DATA	When read: The value written in the most recent successful write access When written: See the description in this section

49.4.10 LIN Receive Register (eSCI_LRR)

This register provides the data bytes of received in case of an LIN RX frame was initiated.

Address: 0h base + 14h offset = 14h

Bit	0	1	2	3	4	5	6	7
Read	D							
Write								
Reset	0	0	0	0	0	0	0	0

eSCI_LRR field descriptions

Field	Description
0–7 D	Receive Data . This field provides the data bytes of received LIN RX frames.

49.4.11 LIN CRC Polynomial Register (eSCI_LPR)

This register provides the CRC polynom for generation and processing of CRC-enhanced LIN frames.

Address: 0h base + 18h offset = 18h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Read	P															
Write																
Reset	1	1	0	0	0	1	0	1	1	0	0	1	1	0	0	1

eSCI_LPR field descriptions

Field	Description
0–15 P	Polynomial bit $x^{P[n]}$. Used to define the LIN polynomial. Reset value results in $x^{15} + x^{14} + x^{10} + x^8 + x^7 + x^4 + x^3 + 1$, which is the polynomial used for the CAN protocol.

49.4.12 Control Register 3 (eSCI_CR3)

This register is used to control the frame formats and the generation of the ERR bit in the [SCI Data Register \(eSCI_SDR\)](#).

Address: 0h base + 1Ah offset = 1Ah

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Read	0				ERO	ERFE	ERPE	M2	0							
Write					E											
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

eSCI_CR3 field descriptions

Field	Description
0–3 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

Table continues on the next page...

eSCI_CR3 field descriptions (continued)

Field	Description
4 EROE	ERR flag overrun enable. 0 eSCI_SDR[ERR] flag not affected by overrun detection. 1 eSCI_SDR[ERR] flag is set on overrun detection during frame reception.
5 ERFE	ERR flag frame error enable. 0 eSCI_SDR[ERR] flag not affected by frame error detection. 1 eSCI_SDR[ERR] flag is set on frame error detection for the data provided in eSCI_SDR.
6 ERPE	ERR flag parity error enable. 0 eSCI_SDR[ERR] flag not affected by parity error detection. 1 eSCI_SDR[ERR] flag is set on parity error detection for the data provided in eSCI_SDR.
7 M2	Frame Format Mode 2. This control bit together with the M bit of the Control Register 1 (eSCI_CR1) controls the frame format used. The supported frame formats and the related settings are defined in Frame Formats .
8–15 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

49.5 Functional Description

This section provides a complete functional description of the eSCI block, detailing the operation of the design from the end user perspective in a number of subsections.

49.5.1 Module Control

The operational mode of the module is controlled by the MDIS bit in the Control Register 2 (eSCI_CR2). The module can transmit and receives data when it is enabled, i.e MDIS=0.

49.5.2 Frame Formats

The eSCI module uses the standard NRZ mark/space data format. The eSCI supports three basic frame types, which are the data frames, break characters, and idle characters.

49.5.2.1 Data Frame Formats

Each data frame contains a character that is surrounded by a start bit, an optional parity or address bit, and one or two stop bits. The supported data frame formats for transmission and reception are specified in [Table 49-19](#). The supported data frame formats for reception only are specified in [Table 49-20](#).

Table 49-19. Supported Data Frame Formats for RX and TX

Control				Frame Content				
eSCI_CR3	eSCI_CR1			Start Bits	Payload Bits			Stop Bits
M2	M	PE	WAKE		Character Bits	Address Bits ¹	Parity Bits	
LIN byte fields (Figure 49-16)								
0	0	0	0	1	8	0	0	1
SCI Frames (8 payload bits)(Figure 49-17)								
0	0	0	0	1	8	0	0	1
0	0	0	1	1	7	1	0	1
0	0	1	0	1	7	0	1	1
SCI Frames (9 payload bits) (Figure 49-18)								
0	1	0	0	1	9	0	0	1
0	1	0	1	1	8	1	0	1
0	1	1	0	1	8	0	1	1

1. The address bit identifies the frame as an address character. See [Multiprocessor Communication](#).

Table 49-20. Supported Data Frame Formats for RX only

Control				Frame Content				
eSCI_CR3	eSCI_CR1			Start Bits	Payload Bits			Stop Bits
M2	M	PE	WAKE		Character Bits	Address Bits	Parity Bits	
SCI Frames (2 stop bits) (see Figure 49-19)								
1	0	1	0	1	8	0	1	2
1	1	1	0	1	12	0	1	2

The structure of the LIN byte fields in normal polarity is shown in [Figure 49-16](#).

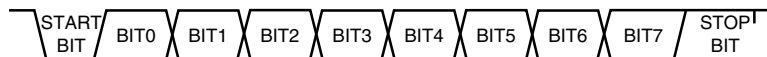


Figure 49-16. LIN Byte Field Format

The structures of the supported SCI frame formats with 8 payload bits are shown in [Figure 49-17](#).

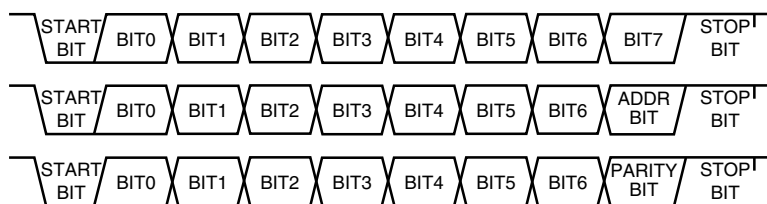


Figure 49-17. SCI Frame Formats (8 payload bits)

The structures of the supported SCI frame formats with 9 payload bits are shown in [Figure 49-18](#).

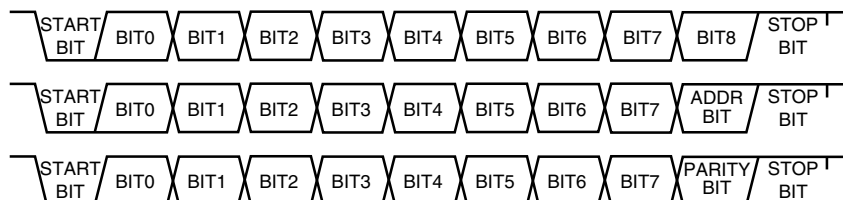


Figure 49-18. SCI Frame Formats (9 payload bits)

The structures of the supported SCI frame formats with 2 stop bits in normal polarity are shown in [Figure 49-19](#). This frame format is supported for reception only.

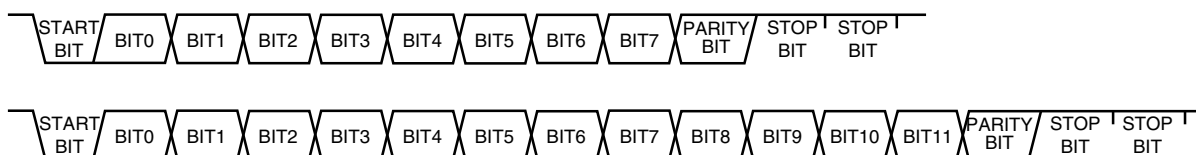


Figure 49-19. SCI Frame Formats (2 stop bits)

49.5.2.1.1 Inverted Data Frame Formats

The structures of the supported data frame formats in inverted polarity are shown in the following figure. These frame types are supported for reception only. The polarity of the RXD pin is controlled by the RXPOL bit in the Control Register 2 (eSCI_CR2).

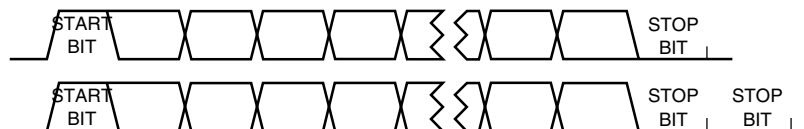


Figure 49-20. Inverted SCI Frame Formats

49.5.2.2 Break Character Formats

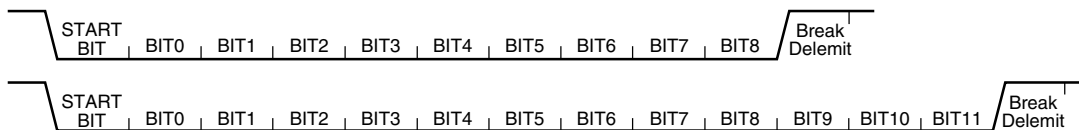
The supported break character formats are specified in [Table 49-21](#).

Table 49-21. Supported Break Character Formats

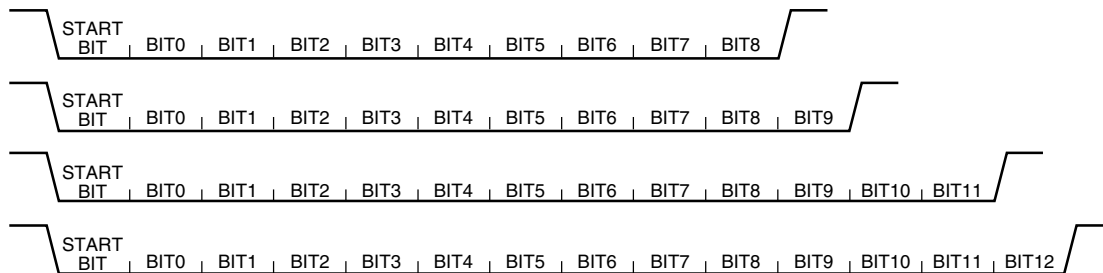
Control ¹			Break Character Content		
eSCI_CR3	eSCI_CR1	eSCI_CR2	Start Bit	Character Bits	Delemit Bits
M2	M	BRCL			
LIN Break Symbol (see Figure 49-21)					
0	0	0	1	9	1
0	0	1	1	12	1
SCI Break Character (see Figure 49-22)					
0	0	0	1	9	0
0	0	1	1	12	0
0	1	0	1	10	0
0	1	1	1	13	0

1. All codings which are not listed are reserved and must not be used.

The structure and content of the LIN break symbols is shown in [Figure 49-21](#).

**Figure 49-21. LIN Break Symbol Format**

The structure and content of the SCI break characters is shown in [Figure 49-22](#).

**Figure 49-22. SCI Break Character Formats**

49.5.2.3 Idle Character Formats

An idle character is a sequence of bits with the value 1. The supported idle character formats are specified in [Table 49-22](#). The preamble has the same structure and content as an idle character.

Table 49-22. Supported Idle Character Formats

Control		Idle Character Length
eSCI_CR3[M2]	eSCI_CR1[M]	
Idle Characters (see Figure 49-23)		
0	0	10
0	1	11
1	0	12
1	1	16

The structure and content of the idle characters is shown in [Figure 49-23](#).

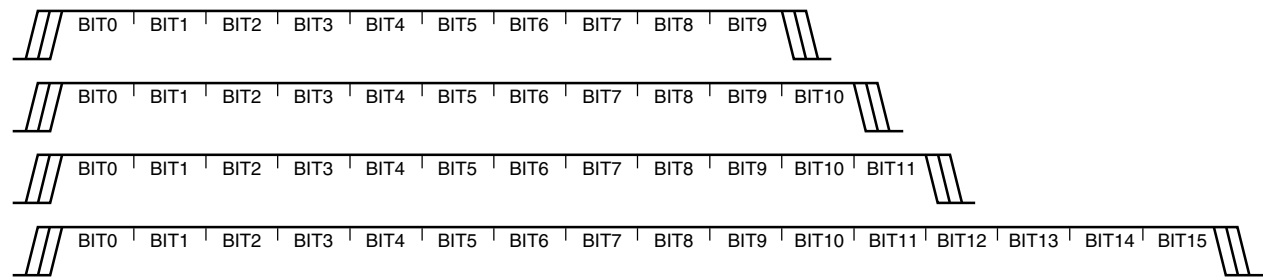


Figure 49-23. Idle Character Formats

49.5.3 Baud Rate and Clock Generation

A 13-bit modulus counter in the baud rate generator derives the baud rate for both the receiver and the transmitter. The value written to the SBR field in the Baud Rate Register (eSCI_BRR) determines the module clock divisor. The baud rate clock is synchronized with the bus clock and drives the receiver. The baud rate clock divided by 16 drives the transmitter. The receiver has an acquisition rate of 16 samples per bit time.

The baud rate generator is enabled when the TE bit or RE bit in the Control Register 1 (eSCI_CR1) is set to 1 for the first time. The baud rate generator is disabled when SBR = 0.

Baud rate generation is subject to one source of error:

- Integer division of the module clock may not give the exact required target baud rate.

The following table lists some examples of achieving target baud rates with a module clock frequency of MCLK = 10.2 MHz.

Table 49-23. Baud Rates Error Example (MCLK = 10.2 MHz)

eSCI_BRR[SBR]	RCLK (Hz)	TCLK (Hz)	Target Baud Rate	Error (%)
17	600,000.0	37,500.0	38,400	2.3
33	309,090.9	19,318.2	19,200	.62
66	154,545.5	9659.1	9600	.62
133	76,691.7	4793.2	4800	.14
266	38,345.9	2396.6	2400	.14
531	19,209.0	1200.6	1200	.11
1062	9604.5	600.3	600	.05
2125	4800.0	300.0	300	.00
4250	2400.0	150.0	150	.00
5795	1760.1	110.0	110	.00

49.5.3.1 Module Clock

For the frequency of module clock MCLK and the system bus clock, see the chip-specific eSCI information. MCLK has the same phase as the system bus clock.

49.5.3.2 Transmitter Clock

The transmitter clock TCLK is used to drive the data to the serial bus via the TXD pin. It is derived from the system bus clock by the baud rate generator. The baud rate generator is controlled by the value of the SBR field in the Baud Rate Register (eSCI_BRR). The frequency of the transmitter clock is determined by the following equation and defines the length of the transmitted bits, which is denoted as the *bit time*.

$$f_{\text{TCLK}} = \frac{f_{\text{MCLK}}}{16 \cdot \text{SBR}}$$

49.5.3.3 Receiver Clock

The receiver clock RCLK is used to sample the data received on the RXD or TXD pin. It is derived from the system bus clock by the baud rate generator. The baud rate generator is controlled by the value of the SBR field in the Baud Rate Register (eSCI_BRR). The frequency of the receiver sample clock is determined by the following equation.

$$f_{\text{RT}} = \frac{f_{\text{MCLK}}}{\text{SBR}}$$

The frequency of the receiver clock is 16 times the frequency of the transmitter clock, thus each bit is sampled with 16 samples. Each of the 16 samples of a bit has a sample number assigned, which is defined by the receiver sample counter RSC. The n-th sample is denoted by RS_n. The receiver sample counter RSC is updated with each rising edge of the receiver clock RCLK.

49.5.4 Baud Rate Tolerance

A transmitting device may be operating at a baud rate below or above the receiver baud rate. Accumulated bit time misalignment can cause one of the three stop bit data samples RS₈, RS₉, and RS₁₀ to fall outside the actual stop bit. A noise error will occur if the stop bit sample RS₈, RS₉, and RS₁₀ samples are not all the same logical value 1. A framing error will occur if the receiver clock is misaligned in such a way that the majority of the RS₈, RS₉, and RS₁₀ stop bit samples are a logic zero.

49.5.4.1 Faster Receiver Tolerance

In this case the receiver has a higher baud rate than the transmitter, thus the stop bit sampling starts already in the last transmitted payload bit. To ensure the correct, noise and framing error free reception of the stop bit, the samples RS₈, RS₉, and RS₁₀ must be located in the transmitted stop bit as shown in the following figure.

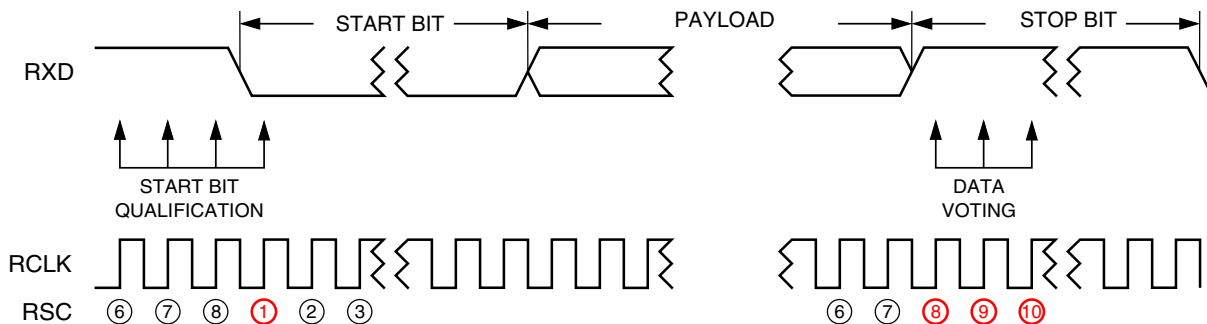


Figure 49-24. Faster Receiver

The maximum tolerance that ensures error free reception can be calculated with the assumption that RS₇ is sampled during the last transmitted payload bit and RS₈ is sampled in the stop bit.

For an frame with n payload bits the transmitter starts the transmission of the stop bit

$$t_{XSTOP} = (n + 1) \cdot 16 \cdot RT_{TR}$$

after the start of the transmission of the start bit.

For an frame with n payload bits the receiver samples the RS8 sample of the stop bit

$$rx_{STOP} = (n + 1) \cdot 16 \cdot RT_{RE} + 7 \cdot RT_{RE}$$

after the successful qualification of the start bit.

To ensure error free reception of the stop bit, the transmitter must start the transmission of the stop bit before the receiver samples RSC8.

$$tx_{STOP} < rx_{STOP}$$

The maximum percent difference between the receiver baud rate and the transmitter baud rate is:

$$\Delta \text{baudrate} \leq \left(\frac{rx_{STOP} - tx_{STOP}}{tx_{STOP}} \right) \times 100$$

The maximum percent differences for the supported frames is given in the following table.

Table 49-24. Faster Receiver Maximum Tolerance

payload bits	max baudrate difference	tx _{STOP}	rx _{STOP}
8	4.63%	144	151
9	4.19%	160	167
13	3.03%	224	231

49.5.4.2 Slower Receiver Tolerance

In this case the receiver has a slower baud rate than the transmitter, thus the stop bit sampling is still running while the next start bit is already transmitted. To ensure the correct, noise and framing error free reception of the stop bit, the samples RS8, RS9, and RS10 must be located in the transmitted stop bit as shown in the following figure.

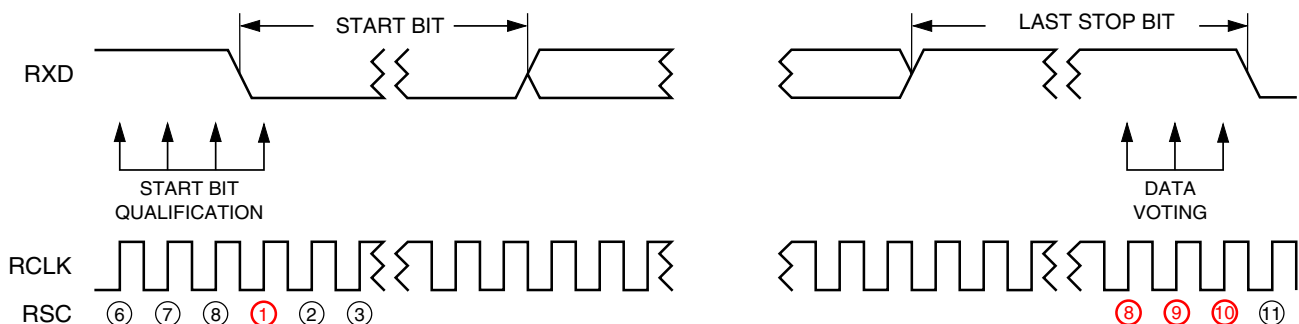


Figure 49-25. Slower Receiver

Functional Description

The maximum tolerance that ensures error free reception can be calculated with the assumption that RS11 is sampled in the transmitted start bit and RS10 is sampled in the last stop bit.

For a frame with n payload bits and s stop bits, the transmitter starts the transmission of the next start bit

$$t_{xSTART} = (n + s + 1) \cdot 16 \cdot RT_{TR}$$

after the start of the transmission of the previous start bit.

For a frame with n payload bits and s stop bits, the receiver samples the RS10 sample of the last stop bit

$$r_{xSTOP} = (n + s) \cdot 16 \cdot RT_{RE} + 9 \cdot RT_{RE}$$

after the successful qualification of the start bit.

To ensure error free reception of the last stop bit, the transmitter must start the transmission of the start bit after the receiver samples RS10.

$$r_{xSTOP} < t_{xSTART}$$

The maximum percent difference between the receiver baud rate and the transmitter baud rate is:

$$\Delta \text{baudrate} \leq \left(\frac{t_{xSTART} - r_{xSTOP}}{t_{xSTART}} \right) \times 100$$

The maximum percent differences for the supported frames is given in the following table.

Table 49-25. Slower Receiver Maximum Tolerance

payload bits	stop bits	max baudrate difference	r_{xSTOP}	t_{xSTART}
8	1	4.37%	153	160
9	1	3.97%	169	176
9	2	3.57%	185	196
13	2	2.73%	249	256

49.5.5 SCI Mode

49.5.5.1 SCI Mode Configuration

The application must configure the following bits and fields in order to achieve correct SCI operation.

- enable *SCI* Mode
 - LIN Control Register 1 (eSCI_LCR1)[LIN]:= 0
- select *baud rate*
 - Baud Rate Register (eSCI_BRR)[SBR]
- select *receiver input* mode
 - Control Register 1 (eSCI_CR1)[LOOPS]
 - Control Register 1 (eSCI_CR1)[RSRC]
- select *frame format*
 - Control Register 1 (eSCI_CR1)[M]
 - Control Register 1 (eSCI_CR1)[PE]
 - Control Register 1 (eSCI_CR1)[WAKE]
 - Control Register 3 (eSCI_CR3)[M2]
- select *parity type*
 - Control Register 1 (eSCI_CR1)[PT]

49.5.5.2 Transmitter

The transmitter supports the transmission of all frame types defined in [Table 49-19](#), of all break characters defined in [Table 49-21](#), and of all idle characters defined in [Table 49-22](#).

49.5.5.2.1 Transmitter States and Transitions

The transmitter has four basic states which are shown and described in [Table 49-26](#). The state transitions that can be triggered by the application commands are shown in [Table 49-27](#). The state transitions that can be triggered by the module are shown [Table 49-28](#). The state diagram of the transmitter is shown in [Figure 49-26](#).

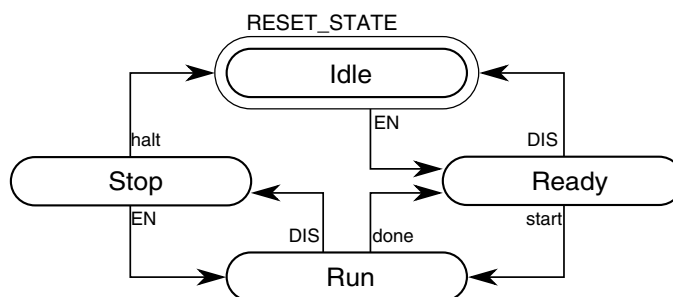


Figure 49-26. Transmitter State Diagram

The current state of the transmitter can be determined by the TE control bit in the Control Register 1 (eSCI_CR1) and the TACT status bit in Interrupt Flag and Status Register 1 (eSCI_IFSR1).

Table 49-26. Transmitter States

State	Indication		Description
	eSCI_CR1[TE]	eSCI_IFSR1[TACT]	
Idle	0	0	Transmitter is <i>disabled</i> and <i>no</i> transmission is running
Ready	1	0	Transmitter is <i>enabled</i> and <i>no</i> transmission is running
Run	1	1	Transmitter is <i>enabled</i> and transmission is running
Stop	0	1	Transmitter is <i>disabled</i> and transmission is running

The application triggers a transition described in [Table 49-27](#) when it issues a command by writing to the TE bit in the Control Register 1 (eSCI_CR1). The transition is triggered only if the conditions are fulfilled. As a result of the transition the state of the transmitter is changed as shown in [Figure 49-26](#) and the action given in [Table 49-27](#) is executed.

Table 49-27. Transmitter Application Transitions

Transition	Command	Precondition	Action	Description
EN	eSCI_CR1[TE]:=1	eSCI_CR1[TE]=0	iPRE:=1	Transmitter is <i>enabled</i> by application command.
DIS	eSCI_CR1[TE]:=0	eSCI_CR1[TE]=1		Transmitter is <i>disabled</i> by application command

The module transition shown in [Table 49-28](#) are triggered when the described condition or event occurs. The send break bit SBK in the Control Register 1 (eSCI_CR1) is checked for the start condition. The internal commit bit iCMT, the transmitter active bit TACT in the Interrupt Flag and Status Register 1 (eSCI_IFSR1), the TDRE, and the TC flag in the Interrupt Flag and Status Register 1 (eSCI_IFSR1) are changed as a action result of the transition.

Table 49-28. Transmitter Module Transitions

Transition	Condition	Action	Description
start	(State=Ready) and (SBK=1 or iPRE=1 or iCMT=1)	TACT:=1	Start of transmission of data frame or special character when data are available or character transmission request is pending.
done	State=Run and last stop bit transmitted	TACT:=0 TC:= (SBK=0 & iPRE=0 & iCMT=0)	Finished transmission of data frame or special character and transmitter still enabled. Transmission is complete if no transmit request is pending.
halt	State=Stop and last stop bit transmitted	TACT:=0 TC:=1 iCMT:=0	Finished transmission of data frame or special character and transmitter was disabled.

49.5.5.2.2 Frame and Character Transmission

The transmitter starts the transmission of a data frame or special character when the condition for the *start* transition as described in [Table 49-28](#) is fulfilled. There are three source for data or character transmission. The priority among these source are specified in the following table. All three sources can be available at one point in time.

Table 49-29. Transmit Source Priority

Priority	Indication	Transmission Source
(highest) 0	iPRE=1	Preamble.
1	eSCI_CR1[SBK]=1	Break character.
(lowest) 2	iCMT=1	SCI Data Register (eSCI_SDR) frame.

49.5.5.2.3 CPU Controlled SCI Data Frame Transmission

The transmission of a data frame is started when the transmitter is in its Ready state and only the commit bit iCMT is set.

As the first step, the content of the SCI Data Register (eSCI_SDR) is transferred into the internal transmitter shift register. When this transfer is finished, the internal commit bit iCMT is cleared and the transmit data register empty flag TDRE in the Interrupt Flag and Status Register 1 (eSCI_IFSR1) is set. If the transmit interrupt enable bit TIE in the Control Register 1 (eSCI_CR1) is also set, the TDRE flag generates a transmitter interrupt request.

The transmitter shift register then shifts a frame out through the TXD output signal, which is prefaced with a start bit and appended with the parity bit, if configured, and the configured number of stop bits.

When the last stop bit has been transmitted and the application has not disabled the transmitter, the transmitter returns to the Ready state via the *done* transition. If no frame or character transmit request is pending, the transfer complete flag TC in the Interrupt Flag and Status Register 1 (eSCI_IFSR1) is set.

If the application has disabled the transmitter while the frame is transmitted and stop bit has been transmitted, the transmitter goes into the Idle state via the *halt* transition. The transfer complete flag TC in the Interrupt Flag and Status Register 1 (eSCI_IFSR1) is set and the internal commit bit iCMT is cleared.

49.5.5.2.4 DMA Controlled SCI Data Frame Transmission

In this mode, the eSCI module handles the generation of Data Frames internally.

When new data is required for transmission, the module generates the transmit DMA request and the DMA controller delivers the required data via write accesses to SCI Data Register (eSCI_SDR). The write access to the low byte of SCI Data Register (eSCI_SDR) triggers the transmission of the data. The write access to the high byte of SCI Data Register (eSCI_SDR) triggers no internal operation.

The application requests the eSCI module to enter this mode by setting the TXDMA bit in the Control Register 2 (eSCI_CR2). From this point in time, the module starts the generation of DMA requests and frame transmission. Before entering this mode, the application should perform the following actions:

1. Configure the module for SCI mode.
2. Enable the transmitter by setting TE in Control Register 1 (eSCI_CR1) to 1.
3. Setup the DMA controller channel and provide frame data in system memory

A block diagram which presents an overview of the DMA Controlled Date Frame Transmission is shown in the following figure.

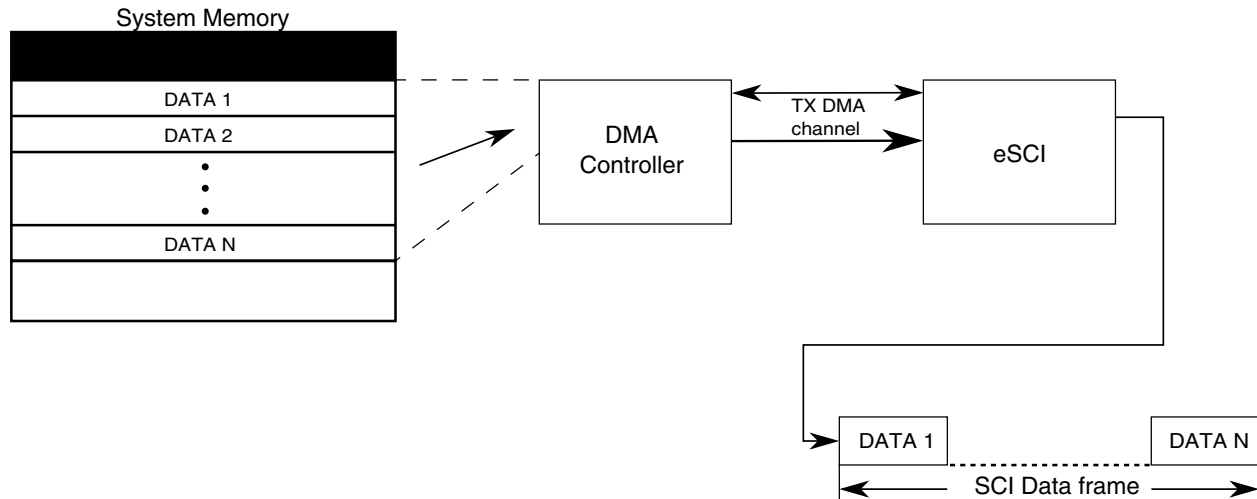


Figure 49-27. DMA Controlled SCI Data Frame generation

49.5.5.2.5 Parity Generation

The eSCI module generates the parity bit in transmitted data frame when the parity enable bit PE in the Control Register 1 (eSCI_CR1) is set. The parity type bit PT in the Control Register 1 (eSCI_CR1) defines whether the odd or even parity is generated.

49.5.5.2.6 Preamble Transmission

The transmission of a preamble is started when the transmitter is in Ready state, the internal iPRE bit, which is not visible to the application, is set, and the SBK in the Control Register 1 (eSCI_CR1) is clear.

After the transmission of the stop bit and if the application has not disabled the transmitter, the transmitter returns to the Ready state via the *done* transition. If no frame or character transmit request is pending, the transfer complete flag TC in the Interrupt Flag and Status Register 1 (eSCI_IFSR1) is set.

If the application has disabled the transmitter while the preamble is transmitted and if the stop bit has been transmitted, the transmitter goes into the Idle state via the halt transition. The transfer complete flag TC in the Interrupt Flag and Status Register 1 (eSCI_IFSR1) is set and the internal commit bit iCMT is cleared.

49.5.5.2.7 Break Character Transmission

The transmission of a break character is started when the transmitter is in Ready state and the send break character bit SBK in the Control Register 1 (eSCI_CR1) is set. After the transmission of the break character and if the application has not disabled the transmitter,

the transmitter returns to the Ready state via the *done* transition and restarts the transmission. As long as SBK bit remains set, the transmitter continues to send break characters.

When the application has cleared the SBK bit or has disabled the transmitter, the transmitter continues to transmit the current break character and after it has finished the transmission of this break character it transmits a stop bit. The stop bit at the end of a break character sequence guarantees the recognition of the start bit of the next data frame.

After the transmission of the stop bit and if the application has not disabled the transmitter, the transmitter returns to the Ready state via the *done* transition. If no frame or character transmit request is pending, the transfer complete flag TC in the Interrupt Flag and Status Register 1 (eSCI_IFSR1) is set.

If the application has disabled the transmitter while the break character is transmitted and if the stop bit has been transmitted, the transmitter goes into the Idle state via the *halt* transition. The transfer complete flag TC in the Interrupt Flag and Status Register 1 (eSCI_IFSR1) is set and the internal commit bit iCMT is cleared.

49.5.5.3 Receiver

The receiver supports the reception of all data frame types defined in [Table 49-19](#) and [Table 49-20](#), of all break character defined in [Table 49-21](#), and of all idle characters defined in [Table 49-22](#).

49.5.5.3.1 Receiver States and Transitions

The receiver has four basic states which are shown and described in [Table 49-30](#). The state transitions that can triggered by the application commands are shown in [Table 49-31](#). The state transitions that can triggered by the module are shown in [Table 49-32](#). The state diagram of the transmitter is shown in the following figure.

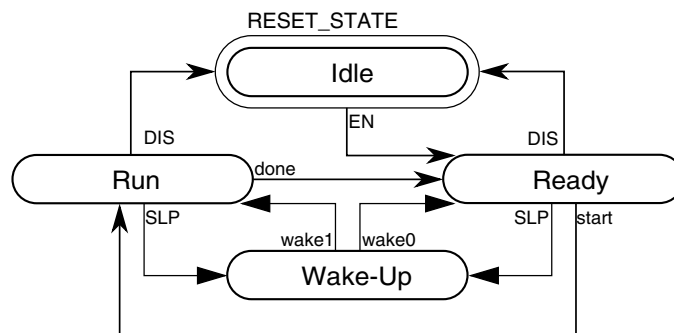


Figure 49-28. Receiver State Diagram

The current state of the receiver can be determined by the RE and RWU bit in the Control Register 1 (eSCI_CR1) and the RACT status bit in Interrupt Flag and Status Register 1 (eSCI_IFSR1).

Table 49-30. Receiver States

State	Indication			Description
	RE	RACT	RWU	
Idle	0	0	0	Receiver is <i>disabled</i> and <i>no</i> reception is running
Ready	1	0	0	Receiver is <i>enabled</i> and <i>no</i> reception is running
Run	1	1	0	Receiver is <i>enabled</i> and reception is running
Wake-up	1	-	1	Receiver is in wake-up mode

The application triggers a transition described in [Table 49-31](#) when it issues a command by writing to the RE bit in the Control Register 1 (eSCI_CR1). The transition is triggered only if the conditions are fulfilled. As a result of the transition, the state of the receiver is changed as shown in the preceeding figure and the action given in [Table 49-31](#) is executed.

Table 49-31. Receiver Application Transition

Transition	Command	Condition	Action	Description
EN	RE:=1	RE=0		Receiver is <i>enabled</i> by application command.
DIS	RE:=0	RE=1		Receiver is <i>disabled</i> by application command
SLP	RWU:=1	RE=1		Receiver is set into wake-up mode

The module transitions shown in [Table 49-32](#) are triggered when the described event occurs.

Table 49-32. Receiver Module Transition

Transition	Condition	Action	Description
start	(State=Ready,Run) and (start bit qualified)	RACT:=1	Start of reception of data frame or break character.
done	(State=Run) and (start bit not verified or idle character received)	RACT:=0	Start Bit not Verified or Idle Character received.
wake0	(State=Wake-up) and (idle character received)	RWU:=0	Wake-up Idle Character received.

Table continues on the next page...

Table 49-32. Receiver Module Transition (continued)

Transition	Condition	Action	Description
wake1	(State=Wake-up) and (address frame received)	RWU:=0	Wake-up address frame received.

49.5.5.3.2 Receiver Input Mode Selection

This section describes the three receiver input modes supported by the eSCI module. The modes are selected by the LOOPS and RSRC control bits in the Control Register 1 (eSCI_CR1).

49.5.5.3.3 Dual Wire Mode

In Dual Wire Mode, the eSCI uses the TXD pin for transmitting and the RXD pin for data receiving.

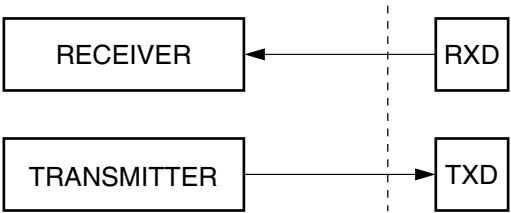


Figure 49-29. Dual Wire Mode

49.5.5.3.4 Single Wire Mode

In Single Wire Mode, the RXD pin is disconnected from the eSCI module and the TXD pin is used for both receiving and transmitting.

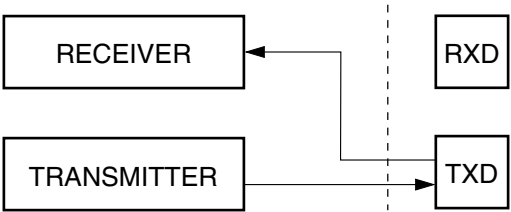


Figure 49-30. Single Wire Mode

In this device, the TXDIR bit (eSCI_CR2[TXDIR]) does not determine whether the TXD pin is going to be used as an input or an output. The relevant pad control register in the System Integration Unit must be programmed for input or output.

49.5.5.3.5 Loop Mode

In Loop Mode, the input of the receiver is driven by the output of the transmitter. The RXD pin is disconnected from the eSCI module.

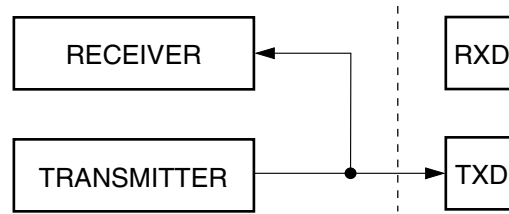


Figure 49-31. Loop Mode

49.5.5.3.6 Frame and Character Reception

The receiver is started when it is in Ready or Wake-up state and on the selected receiver input (see [Receiver Input Mode Selection](#)) an active signal is sampled. The receiver enters the Run or Wake-up state. The received bits are recovered by the bit sampling described in [Bit Sampling](#). During the reception, the received bits are shifted into the internal shift register.

49.5.5.3.7 Break Character Detection

The receiver does not provide any means to detect the reception of a break character. Instead, break characters are processed as data frames. Due to the received 0 at the stop bit location, the reception of a break character causes at least a framing error. The error reporting is performed as described in [Reception Error Reporting](#).

49.5.5.3.8 Idle Character Detection

The Idle character detection starts after the reception of the last stop bit.

49.5.5.3.9 CPU Controlled SCI Data Frame Reception

This section describes the reception process when the receiver is in the Run state.

When the required number of frame bits have been received, the payload bits of the received frame are transferred into SCI Data Register (eSCI_SDR) if the RDRF flag is 0. The receive data register full flag RDRF in Interrupt Flag and Status Register 1 (eSCI_IFSR1) is set. If the receive interrupt enable bit RIE in the Control Register 1 (eSCI_CR1) is set, the RDRF interrupt request is generated.

If an idle character has been detected, the IDLE flag in the Interrupt Flag and Status Register 1 (eSCI_IFSR1) is set. If the idle line interrupt enable bit ILIE in the Control Register 1 (eSCI_CR1) is set, the IDLE interrupt request is generated.

If any of the receiver errors described in [Reception Error Reporting](#) have occurred, that corresponding flags will be set.

If the application is disabled the receiver, by clearing the receiver enable bit RE in the Interrupt Flag and Status Register 1 (eSCI_IFSR1), the current frame is discarded and no flags will be updated.

49.5.5.3.10 DMA Controlled SCI Data Frames Reception

In this mode, the eSCI module controls the reception of SCI Data frames automatically and utilizes the connected DMA channels. A block diagram which presents an overview of the DMA Controlled SCI Data Frame reception is shown in the following figure. The RX DMA channel is used to transfer the received frame data into the memory.

When new data is received, the module generates the receive DMA request and the DMA controller retrieves the provided data from the SCI Data Register (eSCI_SDR). The read access from the low byte of the SCI Data Register (eSCI_SDR) signals the end of the DMA cycle for the current data and triggers the reception of new data. The read access from the SCI Data Register (eSCI_SDR) triggers no internal action.

The application requests the eSCI module to enter this mode by setting the RXDMA bit in the Control Register 2 (eSCI_CR2). From this point in time, the module start the generation of DMA requests and frame transmission and reception. Before entering this mode, the application should perform the following actions:

1. Configure the module for SCI mode.
2. Enable the receiver by setting RE in Control Register 1 (eSCI_CR1) to 1.
3. Setup the DMA controller channel.

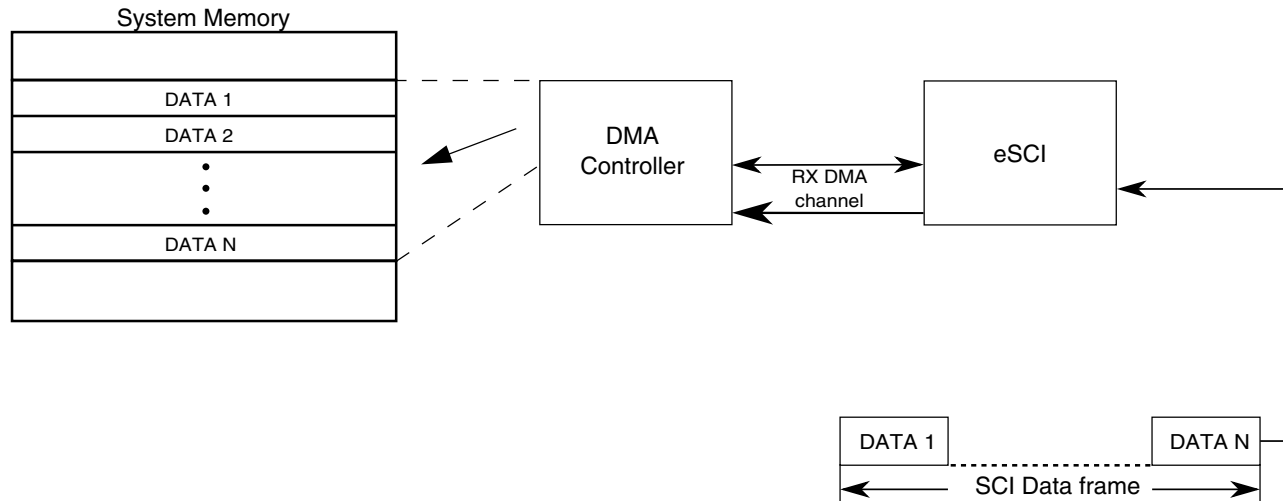


Figure 49-32. DMA Controlled SCI Data Frame Reception

49.5.5.3.11 Receiver Overrun

When the eSCI module has received a frame and attempts to transfer the payload data of the received frame into the SCI Data Register (eSCI_SDR) but neither the application nor the DMA controller hasSCI Data Register (eSCI_SDR) since its last update, the overrun flag OR in the Interrupt Flag and Status Register 1 (eSCI_IFSR1) is set. The data contained in SCI Data Register (eSCI_SDR) are not changed and the received data are lost.

49.5.5.3.12 Wake-up Frame Reception

This section describes the reception process when the receiver is in the Wake-up state.

When the required number of frame bits have been received, the payload bits of the received frame are transferred into SCI Data Register (eSCI_SDR) if the RDRF flag is 0.

If the *address-mark* wake-up mode is selected and the received frame has the address bit set, the receive data register full flag RDRF in Interrupt Flag and Status Register 1 (eSCI_IFSR1) is set. If the receive interrupt enable bit RIE in the Control Register 1 (eSCI_CR1) is set, the RDRF interrupt request is generated. The RWU bit is cleared, and the receiver enters the Run state via the wake1 transition.

If the *idle line* wake-up mode is selected and the receiver has detected an idle character, The RWU bit is cleared, and the receiver enters the Ready state via the wake0 transition.

If any of the receiver errors described in [Reception Error Reporting](#) have occurred, the corresponding flags will be set.

49.5.5.3.13 Bit Sampling

The receiver samples the selected receiver input (see [Receiver Input Mode Selection](#)) with the receiver clock RCLK. The bit sampling for start bit detection is shown in [Figure 49-33](#), the bit sampling for data and stop bit reception is shown in [Figure 49-34](#). The samples indicated by dashed arrows are not used by the receiver. The received data bits are transferred into the internal shift register after the data strobing. If noise or framing errors are detected, this is flagged as described in [Reception Error Reporting](#).

49.5.5.3.14 Bit Synchronization

To adjust for baud rate mismatch, a synchronization of the cyclic receive sample counter RSC is performed during start bit reception as described in [Start Bit Sampling](#).

49.5.5.3.15 Start Bit Sampling

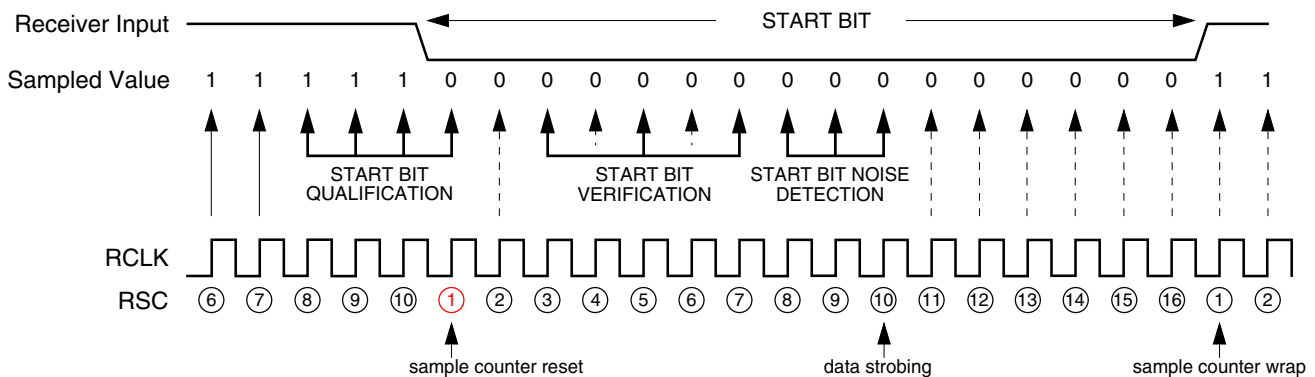


Figure 49-33. Start Bit Sampling and Strobing

The sampling of the start bit consists of three phases, the start bit qualification, the start bit verification, and the start bit noise detection.

49.5.5.3.15.1 Start Bit Qualification

To adjust for baud rate mismatch, the cyclic receive sample counter RSC is re-synchronized after a successful start bit qualification.

A start bit is successfully qualified if

- the start qualification is active, and
- a low sample is read, and
- the low sample was preceded by three consecutive high samples.

The start bit qualification becomes active

- after module reset, or
- after receiver disable and subsequent enable, or
- after the 7-th sample if the start bit verification failed, or
- after the 10-th sample of last stop bit of the preceding frame (example shown in [Figure 49-33](#)).

The start bit qualification becomes inactive

- after successful start bit qualification.

49.5.5.3.15.2 Start Bit Verification

After the successful start bit qualification the receiver starts to verify the start bit by a two out of three samples majority voting.

A start bit is verified if at least two out of the three sample RSC3, RSC5, and RSC7 are sampled low. Noise is detected when exactly one out of the three samples is high. In this case, the noise flag eSCI_IFSR1[NF] is set. The result of the start bit verification is summarized in [Table 49-33](#).

Table 49-33. Start Bit Verification Result

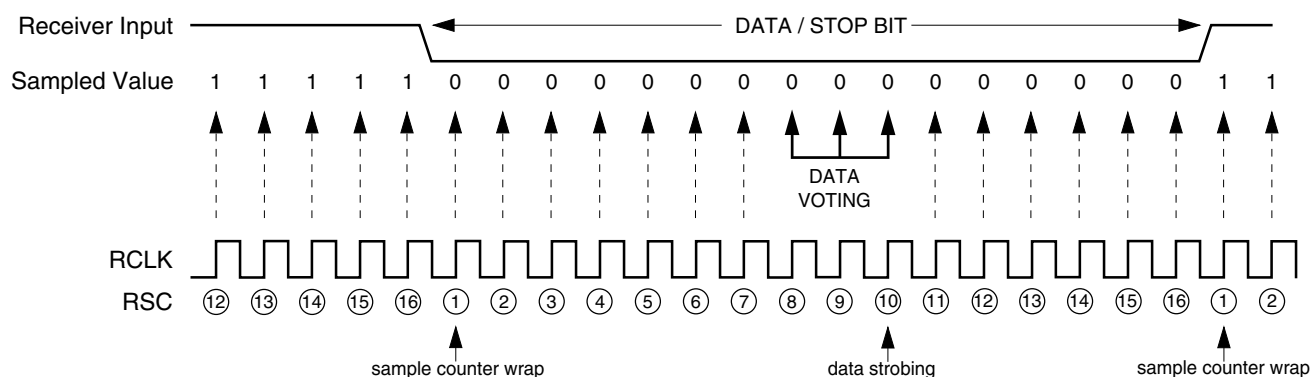
[RSC3, RSC5, RSC7]	Start Bit Verified	Verification Noise Detected
000	Yes	No
001	Yes	Yes
010	Yes	Yes
100	Yes	Yes
011	No	No
101	No	No
110	No	No
111	No	No

If the start bit verification was *not successful*, the receiver activates the start bit qualification. If the start bit verification was *successful*, the receiver continues sampling to perform data noise detection on the samples at RSC8, RSC9, and RSC10. The result of the start bit data noise detection is summarized in [Table 49-34](#). If noise is detected, the noise flag eSCI_IFSR1[NF] is set.

Table 49-34. Start Bit Noise Detection

[RSC8, RSC9, RSC10]	Noise Detected
000	No
001	Yes
010	Yes
100	Yes
011	Yes
101	Yes
110	Yes
111	Yes

49.5.5.3.16 Data Bit Sampling

**Figure 49-34. Data and Stop Bit Sampling and Strobing**

To determine the value of a data bit and to detect noise, a two out of three majority voting is performed on the samples RSC8, RSC9, and RSC10. The following table summarizes the results of the data bit sample. The receiver detects the number of data bit according to the selected frame format. If noise is detected, the noise flag eSCI_IFSR1[NF] is set.

Table 49-35. Data Bit Sampling

[RSC8, RSC9, RSC10]	Data Bit Value	Noise Detected
000	0	No
001	0	Yes
010	0	Yes
100	0	Yes
011	1	Yes
101	1	Yes
110	1	Yes
111	1	No

49.5.5.3.17 Stop Bit Verification

The reception of a valid stop bit is verified if at least two out of the sample RSC8, RSC9, and RSC10 are sampled high. If this is not the case, a framing error is detected. Noise is detected if not all of the samples are of the same value. In this case, the noise flag eSCI_IFSR1[NF] is set. The result of the stop bit verification is summarized in the following table.

Table 49-36. Stop Bit Verification

[RSC8, RSC9, RSC10]	Stop Bit Verified	Framing Error Detected	Noise Detected
000	No	Yes	No
001	No	Yes	Yes
010	No	Yes	Yes
100	No	Yes	Yes
011	Yes	No	Yes
101	Yes	No	Yes
110	Yes	No	Yes
111	Yes	No	No

49.5.5.3.18 Parity Checking

The eSCI module calculates the parity of a received character and checks it versus the received parity bit in the received data frame when the parity enable bit PE in the Control Register 1 (eSCI_CR1) is set. The parity type bit PT in the Control Register 1 (eSCI_CR1) defines whether to check for odd or even parity generation. If a parity error is detected, it is reported as described in [Reception Error Reporting](#).

49.5.5.4 Reception Error Reporting

The receiver can detect four error types: parity errors, framing errors, noise errors, and the overrun error.

The receiver reports the errors detected during frame reception at the end of the reception of the last stop bit of a frame. For error reporting the receiver utilizes the OR, NF, FE, and PF flags in the Interrupt Flag and Status Register 1 (eSCI_IFSR1).

If the receiver has detected an overrun as described in [Receiver Overrun](#), only the OR flag is set. All other error flags are not updated.

If the receiver has detected noise as described in [Bit Sampling](#), the NF flag is set.

If the receiver has *not* detected an overrun and has detected a framing error as described in [Bit Sampling](#), the FE flag is set.

If the receiver has *not* detected an overrun and has detected a parity error as described in [Parity Checking](#), the PF flag is set.

49.5.5.5 Multiprocessor Communication

The multiprocessor communication allows one processor to send blocks of frames to other processors on the same serial link. To avoid the received data interrupt for frames not intended for the processor, the eSCI receiver can be put into the Wake-up state. If the receiver is in the Wake-up state, the eSCI will still load the received data into the SCI Data Register (eSCI_SDR), but will not set the RDRF flag and consequently not request the RDRF interrupt.

The receiver leaves the Wake-up state and clears the RWU bit in the Control Register 1 (eSCI_CR1) when the wake-up pattern configured by WAKE bit in Control Register 1 (eSCI_CR1) is received. The eSCI module supports two types of wake-up patterns, the idle-line wakeup pattern and the address-mark wake-up pattern.

49.5.5.5.1 Idle-Line Wake Up

The idle-line wake-up mode is selected when the WAKE bit in Control Register 1 (eSCI_CR1) is 0. In this mode, the receiver leaves the wake-up state when an idle character is detected as described in [Idle Character Detection](#). The next received frame is the address frame that contains address information which can be evaluated by the application. If the application decides not to receive the frame block, it can set the RWU bit in the Control Register 1 (eSCI_CR1) and return the receiver to the wake-up state.

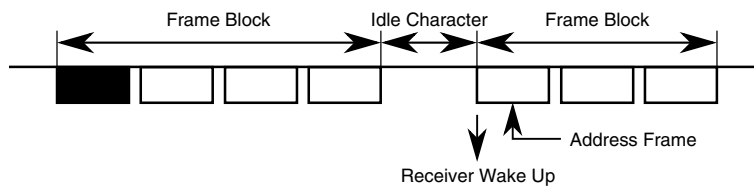


Figure 49-35. Idle-Line Wake Up

49.5.5.5.2 Address-Mark Wake Up

The address-mark wake-up mode is selected when the WAKE bit in Control Register 1 (eSCI_CR1) is 1. If the WAKE bit is set, the address bit is added to the frame format. In this mode, the receiver leaves the wake-up state when a data frame with the address bit value of 1 was received. This frame is the address frame and contains address

information which can be evaluated by the application. If the application decides not to receive the frame block, it can set the RWU bit in the Control Register 1 (eSCI_CR1) and return the receiver to the wake-up state. All data frames that belong to the frame block must have the address bit cleared.

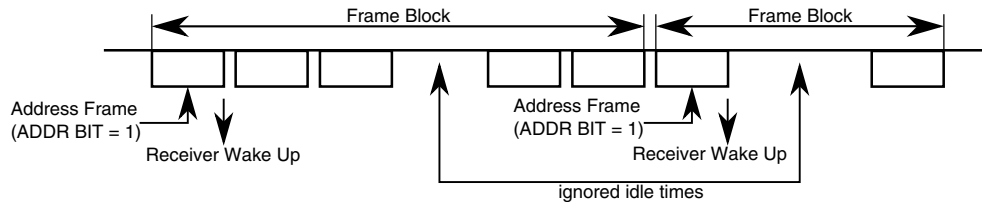


Figure 49-36. Address-Mark Wake Up

49.5.6 LIN Mode

The eSCI provides support for the LIN protocol. It can be used to automate most tasks of a LIN master. In conjunction with the DMA interface it is possible to transmit entire LIN frames and sequences of LIN frames as well as to receive data from LIN slaves without application intervention. There is no special support for LIN slave mode.

49.5.6.1 LIN Mode Configuration

The application must configure the following bits and fields in order to achieve correct LIN operation. The configuration of bits and fields not mentioned in this section depend on the connected LIN slaves and the current application.

- enable *LIN* Mode
 - LIN Control Register 1 (eSCI_LCR1)[LIN]:= 1
- select *RXD* pin as receiver input
 - Control Register 1 (eSCI_CR1)[LOOPS]:= 0
 - Control Register 1 (eSCI_CR1)[RSRC]:= 0
- select *LIN byte fields* as used frame format
 - Control Register 1 (eSCI_CR1)[M]:= 0
 - Control Register 1 (eSCI_CR1)[PE]:= 0
 - Control Register 1 (eSCI_CR1)[WAKE]:= 0
 - Control Register 3 (eSCI_CR3)[M2]:= 0

- select *break character length* of 13 bit as required by LIN 2.0
 - Control Register 2 (eSCI_CR2)[BRCL]:= 1
- select transmission stop on *bit error* detection
 - Control Register 2 (eSCI_CR2)[BESTP]:= 1
- select transmission DMA stop on *bit error* detection
 - Control Register 2 (eSCI_CR2)[BSTP]:= 1
- enable both *transmitter* and *receiver*
 - Control Register 1 (eSCI_CR1)[TE]:= 1
 - Control Register 1 (eSCI_CR1)[RE]:= 1

49.5.6.2 LIN frame formats

The term LIN frame refers to a sequence of LIN byte fields preceded by a break character, both are described in [Frame Formats](#). The eSCI module allows to generate LIN frames for LIN slaves of LIN standards 1.3 and 2.0.

49.5.6.2.1 LIN byte field reception

The reception of a LIN byte field starts with the successful start bit qualification and is finished with the reception of the 16-th sample of the stop bit when no start bit qualification pattern has been detected. If a start bit qualification pattern has been detected at or after the 10-th sample of the stop bit, the reception ends at this sample. An ongoing reception is indicated by the RACT status bit in Interrupt Flag and Status Register 1 (eSCI_IFSR1).

The RACT flag is set if all of the following conditions are fulfilled:

1. the receiver is enabled (eSCI_CR1[RE] = 1), and
2. the LIN task is not in reset (eSCI_LCR1[LRES] = 0), and
3. the start bit qualification pattern has been received (see [Start Bit Sampling](#)).

The RACT flag is cleared if at least one of the following conditions is fulfilled,

1. the receiver is disabled (eSCI_CR1[RE] = 0), or
2. the LIN task is in reset (eSCI_LCR1[LRES] = 1), or

3. the start bit verification fails at sample 7 according to [Table 49-33](#), or
4. the 16-th sample of the stop bit has been received and no start bit qualification pattern has been detected at or after the 10-th sample.

49.5.6.2.2 Standard LIN frames

A standard LIN frame, shown in the following figure consists of a break character, a sync field, an ID field, zero or more data fields, and a checksum field. The data fields and the checksum field are generated by the LIN master for TX LIN frames and generated by the LIN slave for RX LIN frames. The header fields will always be generated by the LIN master.



Figure 49-37. Standard LIN frame format

49.5.6.2.3 CRC Enhanced LIN frames

The CRC Enhanced LIN frames shown in the following figure contain two additional CRC byte fields. These fields are located between the last data field and the Checksum field. The value of the CRC is calculated on the same byte fields as the Checksum is calculated on. The polynom used for the CRC calculation is defined by LIN CRC Polynomial Register (eSCI_LPR). The eSCI module generates the CRC fields for TX frames and checks the CRC fields for RX frames if the CRC bit in the LIN Transmit Register (eSCI_LTR) was written with a value of 1.



Figure 49-38. CRC Enhanced LIN frame format

The CRC Enhanced LIN frames are not part of the LIN standard.

49.5.6.3 LIN TX Frame generation

The eSCI module supports two modes of LIN TX Frame generation, the CPU controlled mode and the DMA controlled mode. In the CPU controlled mode, the application provides the required frame configuration and frame data by subsequent CPU write accesses to the LIN Transmit Register (eSCI_LTR). In the DMA controlled mode, the DMA controller provides the required frame configuration and frame data in response to DMA requests generated by the eSCI module.

49.5.6.3.1 CPU Controlled LIN TX Frame generation

In this mode, the application initiates the generation of an LIN TX Frame and provides the data to be transmitted by a sequence of subsequent CPU write accesses to the LIN Transmit Register (eSCI_LTR). When the eSCI module has processed the data written into LIN Transmit Register (eSCI_LTR), the TXRDY interrupt flag in the Interrupt Flag and Status Register 2 (eSCI_IFSR2) will be set.

The application should clear the TXRDY interrupt flag *before* writing data into the LIN Transmit Register (eSCI_LTR) because the eSCI module will set the TXRDY one clock cycle after the write access.

The first data written to the LIN Transmit Register (eSCI_LTR) provides the Identifier and Identifier Parity fields. The second data written defines the number of data bytes to be transmitted. The third data written defines the CRC and checksum generation. The TD bit has to set to 1 in order to invoke the LIN TX frame generation. The value of the TO field is ignored by the eSCI module for LIN TX frames.

After the third data was written the generation of a LIN TX frame is started. Firstly, a break field is transmitted, then the synch field and the protected identifier field.

All subsequent write accesses to the LIN Transmit Register (eSCI_LTR) provide data bytes to be transmitted via the LIN bus. A data byte field will be transmitted as soon as data are available. After the last data byte, defined by the value written to the LEN field, was send out, the configured CRC and checksum fields will be send out.

After the transmission of the checksum field of the LIN TX frame, the write access counter for the LIN Transmit Register (eSCI_LTR) is reset and the FRC interrupt flag in the Interrupt Flag and Status Register 2 (eSCI_IFSR2) is set.

49.5.6.3.2 DMA Controlled LIN TX Frame generation

In this mode, the eSCI module controls the generation of an LIN TX Frame. When new data is required for transmission, the eSCI module generates the transmit DMA request and the DMA controller delivers the required data. The application requests the eSCI module to enter this mode by setting the TXDMA bit in the Control Register 2 (eSCI_CR2). From this point in time, the module starts the generation of DMA requests and initiates the frame transmission. Before entering this mode, the application should perform the following actions:

1. Configure the module for LIN mode.
2. Enable the transmitter by setting TE in Control Register 1 (eSCI_CR1) to 1.
3. Setup the DMA controller channel and provide frame data in system memory.

A block diagram which presents an overview of the DMA Controlled LIN TX Frame is shown in the following figure. The content of the fields in the memory is the same as described in LIN Transmit Register (eSCI_LTR) - LIN TX frame generation.

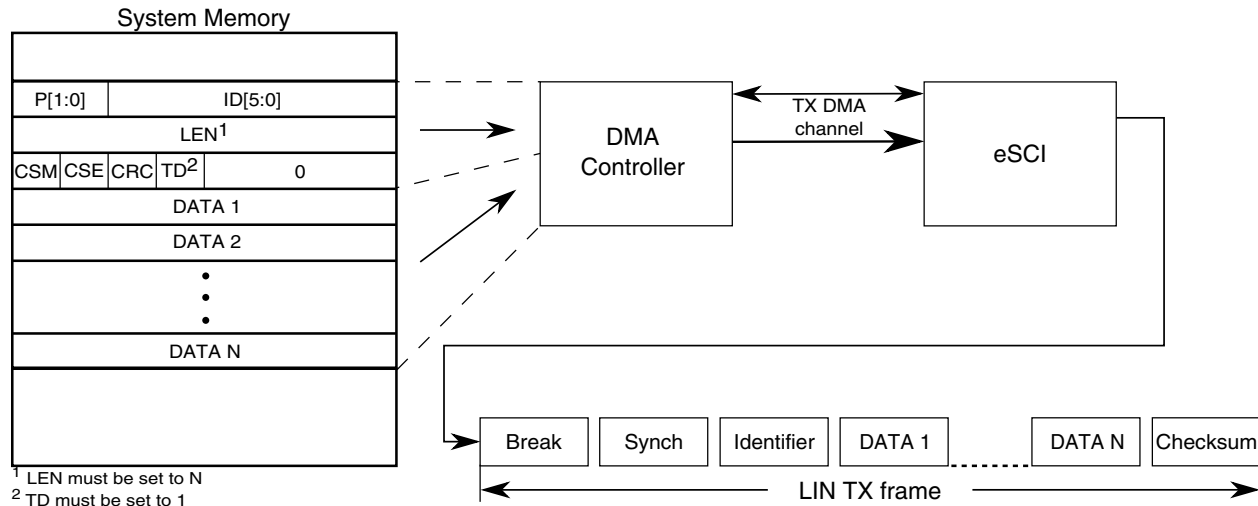


Figure 49-39. DMA Controlled LIN TX Frame generation

49.5.6.4 LIN RX frame generation

The eSCI module supports two modes of LIN RX Frame generation and reception, the CPU controlled mode and the DMA controlled mode. In the CPU controlled mode, the application provides the required data by subsequent CPU write accesses to the LIN Transmit Register (eSCI_LTR) and retrieves the received data by subsequent CPU read accesses to the LIN Receive Register (eSCI_LRR). In the DMA controlled mode, the DMA controller provides the required frame configuration data in response to DMA requests generated by the eSCI module and transfers the received frame data to the memory in response to DMA requests generated by the eSCI module.

49.5.6.4.1 CPU Controlled LIN RX Frames generation

In this mode, the application initiates the generation of a LIN RX Frame by a sequence of subsequent CPU write accesses to the LIN Transmit Register (eSCI_LTR). When the eSCI module has processed the data written into LIN Transmit Register (eSCI_LTR), the TXRDY interrupt flag in the Interrupt Flag and Status Register 2 (eSCI_IFSR2) will be set.

The application should clear the TXRDY interrupt flag *before* writing data into the LIN Transmit Register (eSCI_LTR) because the eSCI module will set the TXRDY one clock cycle after the write access.

The first data written to the LIN Transmit Register (eSCI_LTR) provides the Identifier and Identifier Parity fields. The second data written defines the number of data bytes requested from the LIN slave. The third data written defines the CRC and checksum generation. The TD bit has to set to 0 to invoke the RX frame generation. The TO field defines the upper part of the timeout value. The fourth byte written defines the lower part of the timeout value.

After the fourth byte was written the generation of a LIN RX frame is started. Firstly, a break field is transmitted, then the synch field and the protected identifier field. After the transmission of the protected identifier, the eSCI module starts to receive the frame data transmitted by the LIN slave. When the module has received a complete byte field, the received data are transferred into the LIN Receive Register (eSCI_LRR) and the receive data ready flag RXRDY in the Interrupt Flag and Status Register 2 (eSCI_IFSR2) is set.

The application can retrieve the received data by subsequent read access from LIN Receive Register (eSCI_LRR) after checking the RXRDY flag. The application should clear the RXRDY flag immediately after reading the LIN Receive Register (eSCI_LRR).

After the reception of the configured number of data from the slave, the module starts the reception of the configured CRC and Checksum byte fields. These data are not transferred into the LIN Receive Register (eSCI_LRR). The CRC and Checksum checking is performed internally. In case of errors, they will be reported as described in [LIN Error Reporting](#).

After the reception of the checksum field of the LIN RX frame, the FRC interrupt flag in the Interrupt Flag and Status Register 2 (eSCI_IFSR2) is set.

49.5.6.4.2 DMA Controlled LIN RX Frames generation

In this mode, the eSCI module controls the generation of LIN RX frame header and the reception of the frame data automatically and utilizes the two connected DMA channels. A block diagram which presents an overview of the DMA Controlled LIN RX Frame generation and reception is shown in [Figure 49-40](#). The content of the header fields in the memory is the same as described in LIN Transmit Register (eSCI_LTR) - LIN RX frame generation. The TX DMA channel is used to fetch the LIN RX frame header and control information. The RX DMA channel is used to transfer the received frame data into the memory.

When new data is required for transmission, the module generates the transmit DMA request and the DMA controller delivers the required data. When new data is received, the module generates the receive DMA request and the DMA controller retrieves the provided data.

The application requests the eSCI module to enter this mode by setting the RXDMA bit in the Control Register 2 (eSCI_CR2). From this point in time, the module starts the generation of DMA requests and frame transmission and reception. Before entering this mode, the application should perform the following actions:

1. Configure the module for LIN mode.
2. Enable transmitter and receiver by setting TE and RE in Control Register 1 (eSCI_CR1) to 1.
3. Setup the two DMA controller channels and provide frame header data in system memory.

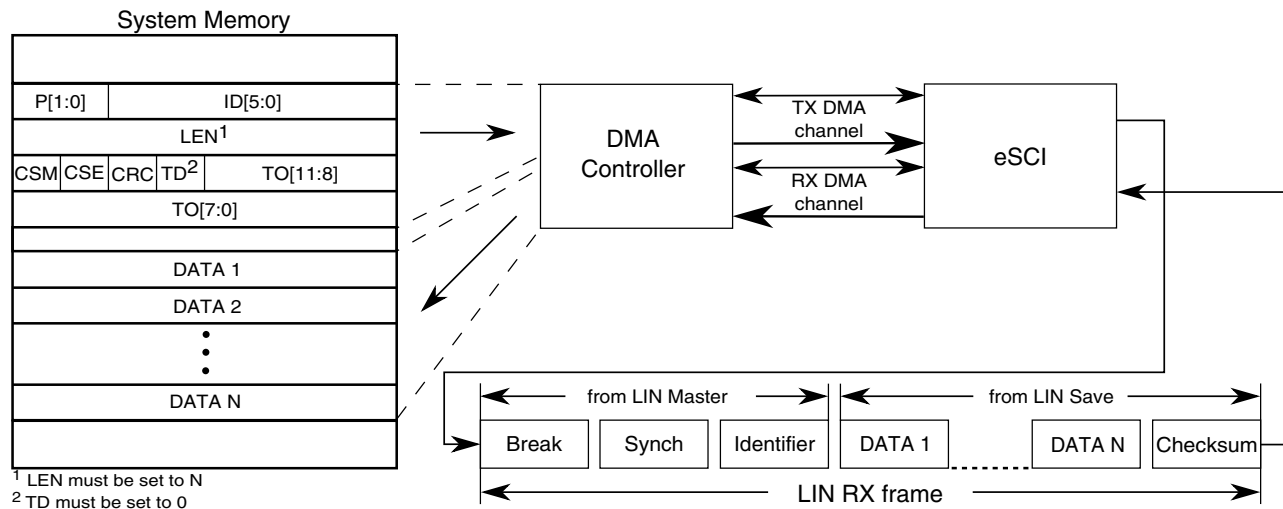


Figure 49-40. DMA Controlled LIN RX Frame generation and reception

49.5.6.5 LIN Error Reporting

This section describes error checking and the signaling of detected errors in LIN mode.

49.5.6.5.1 Physical Bus Error Detection

If the receiver input is sampled 0 for at least 31 sample clock cycles after the start of the transmission of a LIN frame, the physical bus error flag PBERR in the Interrupt Flag and Status Register 2 (eSCI_IFSR2) will be set.

49.5.6.5.2 Unrequested Activity Detection

If an unrequested byte is received (i.e. a byte which is not part of an RX frame) which is not recognized as a wake-up or break character, the bit error flag BERR in the Interrupt Flag and Status Register 1 (eSCI_IFSR1) is set. In addition the RXRDY flag will also be set, the LINRX register must be read before normal operations can proceed.

49.5.6.5.3 Standard Bit Error Detection

The standard bit error detection is enabled when the fast bit error detection control bit FBR in the Control Register 2 (eSCI_CR2) is 0. The standard bit error detection is performed after each LIN byte field transmission.

During the transmission of the LIN frame header and LIN frame data, the receiver is running and receives the signal values on the serial bus. After the complete transmission and the related reception of a LIN byte field, the eSCI compares the data that was transmitted and the data that has been received. If they do not match, the bit error interrupt flag BERR in the Interrupt Flag and Status Register 1 (eSCI_IFSR1) is set.

49.5.6.5.4 Fast Bit Error Detection

Fast Bit Error Detection has been designed to allow flagging of LIN bit errors while they occur, rather than flagging them after a byte transmission has completed (see [Figure 49-41](#)).

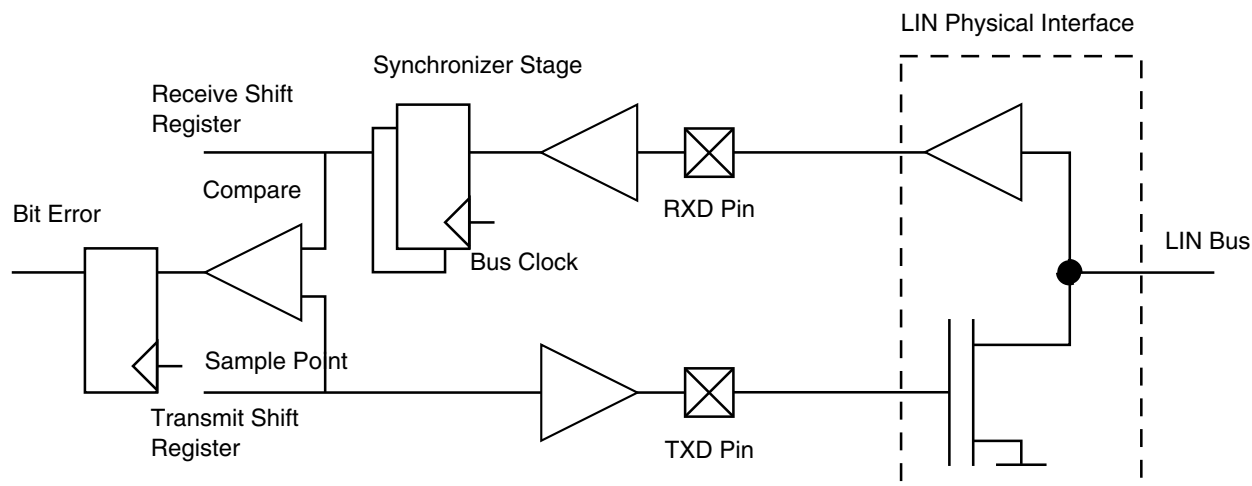


Figure 49-41. Fast Bit Error Detection on a LIN Bus

If fast bit error detection bit FBR in the Control Register 2 (eSCI_CR2) is set, the eSCI will compare the transmitted and the received data stream while the transmitter is active (not idle). Once a mismatch between the transmitted data and the received data is detected, the bit error flag BERR will be set.

To adjust to different bus loads the sample point at which the incoming bit is compared to the one which was transmitted can be selected with the BESM bit in the Control Register 2 (eSCI_CR2). If eSCI_CR2[BESM] = 1, the comparison will be performed with sample RS13, otherwise with RS9 (see [Figure 49-42](#)) (also see [Bit Sampling](#)).

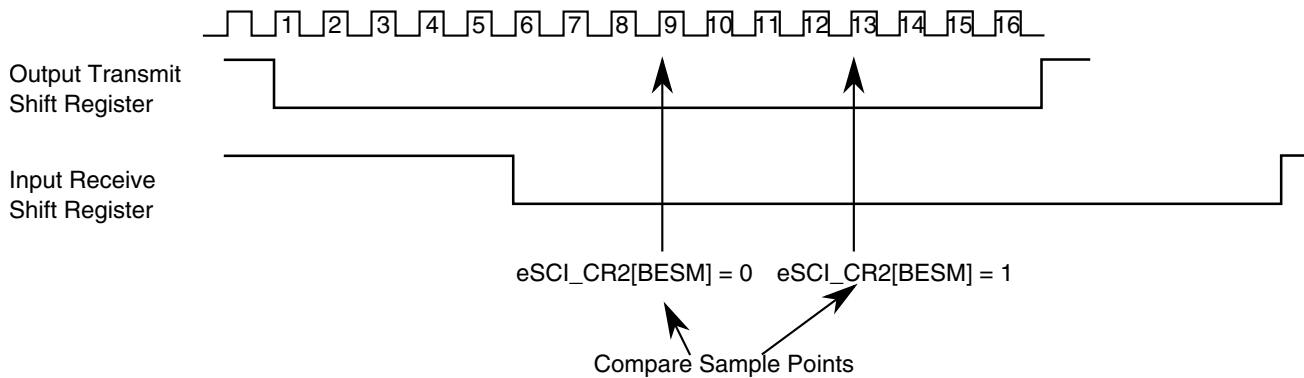


Figure 49-42. Timing Diagram Fast Bit Error Detection

Note

To calculate the exact position of the sample point with regard to the RX pin, the delays through the pads and the two Bus Clock cycle delay through the input synchronizer also needs to be taken into account.

49.5.6.5.5 Slave-Not-Responding-Error Detection

The Slave-Not-Responding-Error is defined in [LIN Specification Package Revision 1.3; December 12, 2002](#); 6 ERROR AND EXCEPTION HANDLING. The LIN specification requires that a NO_RESPONSE_ERROR has to be detected if a message frame is not fully completed within the maximum length $T_{\text{FRAME\_MAX}}$ by any slave task upon transmission of the SYNCH and IDENTIFIER fields. The maximum frame length $T_{\text{FRAME\_MAX}}$ is defined in [LIN Specification Package Revision 1.3; December 12, 2002](#); 3.3 LENGTH OF MESSAGE FRAME AND BUS SLEEP DETECT, as

$$T_{\text{FRAME\_MAX}} = (10 \cdot N_{\text{DATA}} + 45) \cdot 1.4$$

where N_{DATA} is the number of data byte fields of the message frame.

The STO interrupt flag in the Interrupt Flag and Status Register 2 (eSCI_IFSR2) will be set, if an LIN RX frame was not fully received in the amount of time specified in the timeout value field TO in the LIN Transmit Register (eSCI_LTR). The time period starts with the falling edge of the transmitted LIN break character and is specified in units of transmit bits.

To achieve LIN compliant Slave-Not-Responding-Error detection, the timeout value TO in the LIN Transmit Register (eSCI_LTR) field has to be set to $T_{\text{FRAME\_MAX}}$ when a LIN RX frame is initiated.

49.5.6.5.6 Checksum Error Detection

If the checksum enable bit CSE in the LIN Transmit Register (eSCI_LTR) was set, the checksum checking is performed based on the received checksum byte. The checksum mode is selected by the CSM bit in the LIN Transmit Register (eSCI_LTR). If the value received in the checksum bytes did not match the calculated checksum, the checksum error flag CKERR in the Interrupt Flag and Status Register 2 (eSCI_IFSR2) will be set.

49.5.6.5.7 CRC Error Detection

The CRC checking is performed on the two received CRC bytes CRC1 and CRC2 if the CRC Enhanced LIN frame format was selected by the CRC bit in the LIN Transmit Register (eSCI_LTR). If the value received in the two CRC bytes did not match the calculated CRC pattern, the CRC error flag CERR in the Interrupt Flag and Status Register 2 (eSCI_IFSR2) will be set.

49.5.6.5.8 Overflow Detection

When the receiver has received the next byte field, which should be transferred into the LIN Receive Register (eSCI_LRR), but neither the application nor the RX DMA channel have read data from this register since the last update, the received data overflow flag OVFL in the Interrupt Flag and Status Register 2 (eSCI_IFSR2) will be set. In this case the content of the LIN Receive Register (eSCI_LRR) is not changed. The data received most recently are lost.

49.5.6.6 LIN Wake Up

The section describes the LIN Wake Up behavior of the eSCI module.

49.5.6.6.1 LIN Wake-Up Request Generation

The eSCI module can cause the LIN bus to exit the sleep mode by sending a wake-up signal frame, which consists of a wake-up signal 0x80 (consisting of 8 dominant bits followed by 1 recessive bit), followed by the wake-up delimiter period as defined by the WUD field in the LIN Control Register 1 (eSCI_LCR1), followed by Idle time which is n bits, where $n \geq 0$. This Idle time depends on the time until next frame is started by the software.

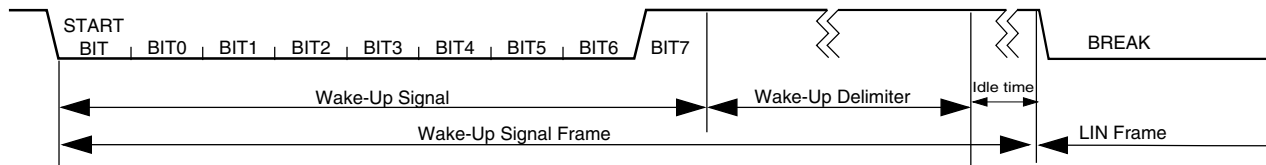


Figure 49-43. LIN Wake-Up Signal Frame

The application triggers the transmission of a wake-up signal frame by writing 1 to the LIN bus wake-up trigger WU in the LIN Control Register 1 (eSCI_LCR1).

The LIN Specification 2.0 requires the generation of LIN wake-up signals as dominant pulses longer than 250 μ s and shorter than 5 ms. To achieve this, the eSCI module has to be programmed to a baud rate between 32 kBaud and 1.6 kBaud. With each of these baud rate settings, the wake-up signal is transmitted as a dominant pulse longer than 250 μ s and shorter than 5 ms.

49.5.6.6.2 LIN Wake-Up Request Detection

The eSCI module detects a LIN wake-up requests when

1. one of the characters 0x00, 0x80, or 0xC0 has been received,
2. followed by zero or more low bits,
3. followed by at least one high bit, and
4. no LIN frame transmission or reception is started or running during the reception above

If a LIN wake-up request has been detected, the LIN wake-up flag LWAKE in the Interrupt Flag and Status Register 2 (eSCI_IFSR2) will be set after the reception of the first high bit.

The LIN Specification 2.0 requires the detection of LIN wake-up requests as dominant pulses longer than 150 μ s. To achieve this, the eSCI module has to be programmed to the maximum baud rate that is not greater than 43.77 kBaud. With this baud rate setting, any dominant pulse longer than 150 μ s is decoded as at least 7 dominant bits (one start and 6 data bits) and consequently as one of the characters 0xC0, 0x80, or 0x00.

49.5.6.7 LIN Protocol Engine Stop and Reset

The LIN protocol engine is stopped and reset when the application sets the LRES control bit in the LIN Control Register 1 (eSCI_LCR1) to 1. In this case, the LIN protocol engine will stop immediately. No new transmissions or receptions are initiated, the LIN serial

bus is driven with the recessive value 1. Additionally to the stop and reset of the LIN protocol engine the receiver and transmitter modules are stopped and reset as well, and the receive and transmit DMA requests are deasserted.

In order to start the LIN Protocol Engine with idle transmitter and receiver processes, the LRES bit should be asserted until all of the status bits DACT, WACT, LACT, TACT, and RACT in the Interrupt Flag and Status Register 1 (eSCI_IFSR1) are cleared. These status bits are cleared within one bit time after assertion of the LRES bit.

49.5.7 Interrupts

This section describes the interrupt sources and interrupt request generation.

49.5.7.1 Interrupt Flags and Enables

All interrupt sources, interrupt flags, and interrupt enable bits are listed in the following table. This table indicates the operational modes, where the interrupt flags can be set by the eSCI module.

Table 49-37. eSCI Interrupt Flags and Interrupt Enable Bits

Interrupt Source	Operational Mode	Interrupt Flag	Interrupt Enable Bit
Transmitter	SCI	eSCI_IFSR1[TDRE]	eSCI_CR1[TIE]
Transmitter	SCI, LIN	eSCI_IFSR1[TC]	eSCI_CR1[TCIE]
Receiver	SCI	eSCI_IFSR1[RDRF]	eSCI_CR1[RIE]
Receiver	SCI	eSCI_IFSR1[IDLE]	eSCI_CR1[ILIE]
Receiver	SCI	eSCI_IFSR1[OR]	eSCI_CR2[ORIE]
Receiver	SCI, LIN	eSCI_IFSR1[NF]	eSCI_CR2[NFIE]
Receiver	SCI, LIN	eSCI_IFSR1[FE]	eSCI_CR2[FEIE]
Receiver	SCI	eSCI_IFSR1[PF]	eSCI_CR2[PFIE]
Receiver	LIN	eSCI_IFSR1[BERR]	eSCI_CR2[BERRIE]
Receiver	LIN	eSCI_IFSR2[RXRDY]	eSCI_LCR1[RXIE]
Transmitter	LIN	eSCI_IFSR2[TXRDY]	eSCI_LCR1[TXIE]
Receiver	LIN	eSCI_IFSR2[LWAKE]	eSCI_LCR1[WUIE]
Receiver	LIN	eSCI_IFSR2[STO]	eSCI_LCR1[STIE]
Receiver	LIN	eSCI_IFSR2[PBERR]	eSCI_LCR1[PBIE]
Receiver	LIN	eSCI_IFSR2[CEERR]	eSCI_LCR1[CIE]
Receiver	LIN	eSCI_IFSR2[CKERR]	eSCI_LCR1[CKIE]
Receiver	LIN	eSCI_IFSR2[FRC]	eSCI_LCR1[FCIE]
Receiver	LIN	eSCI_IFSR2[UREQ]	eSCI_LCR2[URIE]
Transmitter, Receiver	LIN	eSCI_IFSR2[OVFL]	eSCI_LCR2[OFIE]

49.5.7.2 Interrupt Request Generation

The eSCI module provides one hardware interrupt request signal to the systems interrupt controller. This interrupt request signal is asserted if and only if at least one of the interrupt flags and the corresponding interrupt enables are set to 1. Otherwise the interrupt line is deasserted.

49.6 Application Information

49.6.1 SCI Data Frames Separated by Preamble

To separate SCI data frame with preambles with minimum idle line time, use this sequence between messages:

1. write to SCI Data Register (eSCI_SDR)
 - this sets the internal iCMT bit which requests the data transmission
2. wait until TDRE in Interrupt Flag and Status Register 1 (eSCI_IFSR1) is set
 - this indicates the start of transmission; the iCMT bit was cleared
3. clear and subsequently set the TE bit in Control Register 1 (eSCI_CR1)
 - this sets the internal iPRE bit which requests the preamble transmission
4. write to SCI Data Register (eSCI_SDR)
 - this sets the internal iCMT bit which requests the data transmission

The priority scheme of the transmitter which is described in [Table 49-29](#) ensures that the preamble is transmitted before the data frame.

Chapter 50

Resets

50.1 Overview

This device supports the following system reset sources:

- Power-on reset (at power up)
- External reset (from RESET pin)
- Fault Collection and Control Unit (FCCU) reset
- FCCU Output Supervision Unit (FOSU) reset
- Dual core reset
- JTAG reset
- Self Test Control Unit (STCU) reset
- Software system reset
- Life cycle manipulation reset

A reset event caused by any of the above sources will trigger a reset cycle with the assertion of internal reset signals as well as the assertion of the RSTOUT pin. Software may set the SIU_SRCR[SER] bit to 1 to assert the RSTOUT pin without causing an internal reset cycle. This action is referred to as a software external reset. After the reset cycle completes, software may learn the source of the last reset cycle by looking at the Reset Status Register (SIU_RSR) in the SIU. The SIU_RSR register is updated for all reset sources except JTAG and life cycle manipulation reset.

Power-on reset is the most comprehensive form of reset. It affects all modules without exception. Other sources of reset do not affect special hardware that needs to be active during the reset cycle, like for example the state machine of the reset controller. Some modules are affected by all reset sources, but are released from reset state before other modules during the execution of the reset cycle. This reset releasing sequence allows device configuration data to be read from flash memory and applied to the system before the completion of the reset cycle and prior the cores starting code execution.

When the reset cycle completes, the RSTOUT pin is negated. In parallel, the CSE security engine starts execution of the secure boot, and Core 0 starts execution. Core 1 and the associated checker core are kept in reset state until software running on Core 0 configures its reset vector and enables it to start code execution. Reset vectors and core resets are configured by the SIU_RSTVEC0 and SIU_RSTVEC1 registers.

The reset cycle can be extended by the assertion of the RESET pin. This is useful to allow JTAG communication to be established during the reset cycle, so that debugging configuration may be done before the cores start executing.

50.2 Reset sequence

The following figure illustrates the reset flow.

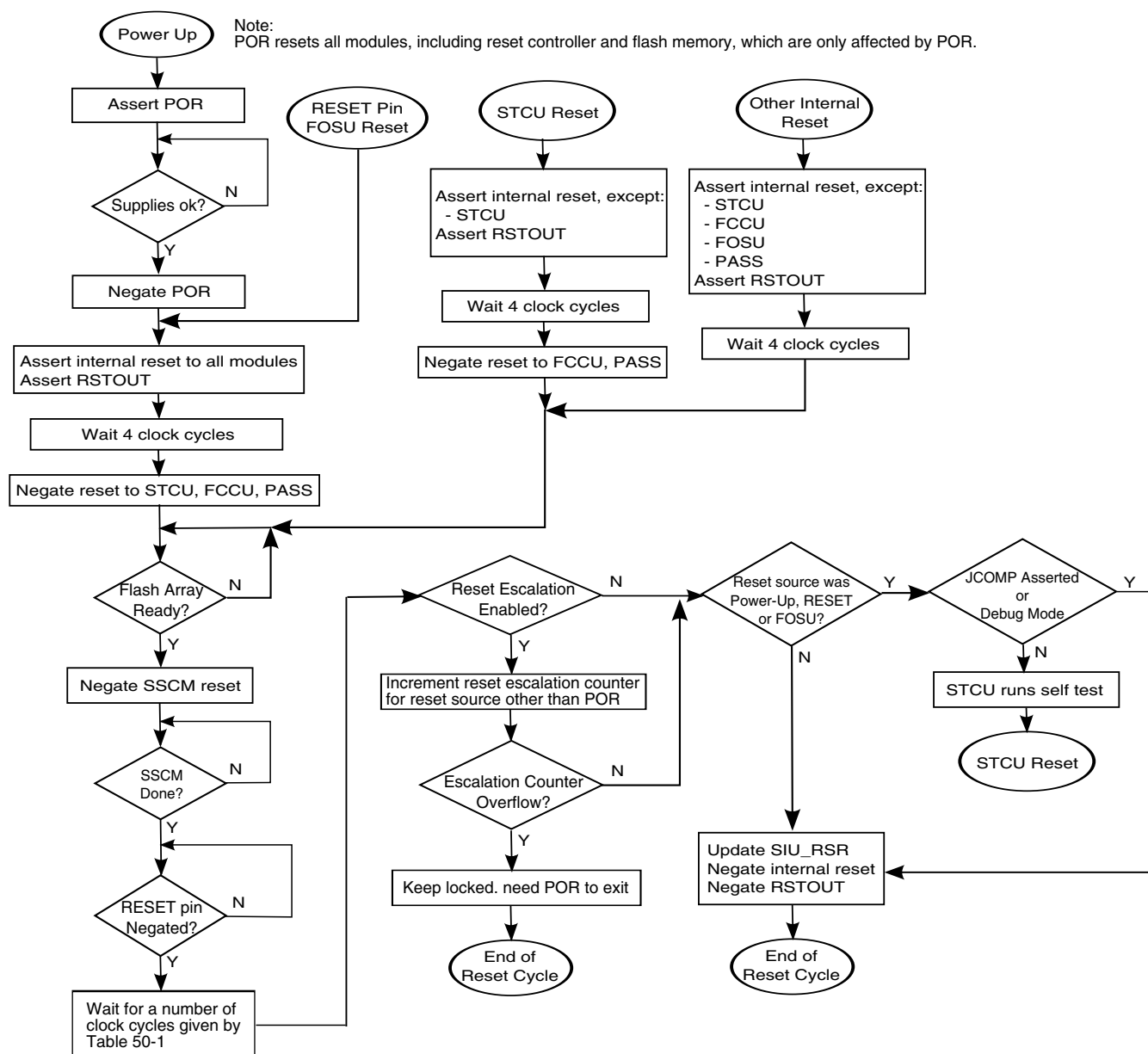


Figure 50-1. Internal reset flow diagram

When power-up is the source of the reset cycle, the following sequence of events takes place:

1. A power-on reset (POR) signal resets everything while power supplies are ramping up.
2. When supply rails are within acceptable (yet still untrimmed) voltage ranges, the POR signal is negated, the RSTOUT pin is asserted, and an internal reset signal is asserted to all modules.

NOTE

Some components are exceptions to this internal reset assertion. They include the reset controller, flash memory, and DCF clients (ERROR function and Reset escalation), all of which are affected only by POR.

3. The reset controller waits for 4 clock cycles so that the reset signal can propagate through the registers, then it negates the reset signals going to the STCU, FCCU and PASS, because these modules need to be active during the reset cycle.
4. The reset controller waits until the flash memory finishes its initialization, then negates the reset signal going to the SSCM module.
5. The SSCM reads device configuration and trimming data from DCF records stored in flash memory and applies the configuration data to the system. Refer to the DCF chapter for more information.
6. When the SSCM is done, the reset controller waits until the RESET pin is negated and then starts a timer that counts for a pre-determined number of clock cycles, which depends on the reset source (see [RSTOUT](#)).
7. If reset escalation is enabled, the reset cycle proceeds with reset escalation evaluation. A reset escalation counter is incremented at the end of every non-POR reset cycle. If the counter exceeds the number programmed in the RET field of SIU_RCR, then the device is locked permanently into a reset state which can only be undone by a power-up cycle. To avoid reaching the reset escalation threshold, software must reset the escalation counter by servicing the SIU_RCR register (write something to the RET field; writing zero disables the reset escalation process).
8. If the device was not locked at reset state by the reset escalation process, the STCU may take control of clocks and resets of the device to execute self-test diagnostics. Self-test is executed during the reset cycle when the following conditions are satisfied:
 - The reset cycle was initiated from power-up or RESET pin or FOSU reset
 - STCU execution is enabled by DCF configuration
 - JCOMP pin is not asserted and debug mode is not engaged

After the self-test is finished, the STCU records the test results and then triggers a new internal reset cycle via an STCU reset request. This new reset cycle will not trigger a self-test execution.

9. If the STCU did not execute, then the reset cycle reaches its normal termination flow, the SIU_RSR register is updated, the RSTOUT pin is negated, and the internal reset signals are negated.

Reset cycles initiated by sources other than power-up, RESET pin, or FOSU do not affect all modules. For a reset cycle initiated by the STCU, the STCU itself is not reset, so that diagnostic results from the previous reset cycle are preserved.¹ Other sources of internal reset will not reset the STCU, FCCU, FOSU and PASS modules (they have information that needs to be preserved through the reset cycle).

50.3 Reset pins

50.3.1 RESET

The RESET pin is an active low input. The RESET pin must be asserted by an external device during power-up or whenever an external reset is required.

The RESET pin has hysteresis, and the hysteresis is always enabled.

The reset controller provides a glitch detector connected to the RESET pin. If the reset controller detects that the RESET pin is asserted for more than two peripheral clock cycles, the event is latched. After the latch is set, if the RESET pin is negated before 10 clock cycles is reached, the reset controller sets the RGF bit without affecting any of the other bits in the reset status register (SIU_RSR). The latch is cleared when the RGF bit is set or a valid reset is recognized. The RGF bit remains set until cleared by a software or the RESET signal asserts for 10 clock cycles.

50.3.2 RSTOUT

The RSTOUT pin is an active low output that uses a push/pull configuration. The RSTOUT pin is driven low by the MCU for all internal and external reset sources.

The time interval between RESET pin negation and RSTOUT pin negation is deterministic, but it depends on the reset source, as shown in the following table. The actual time interval is the value in [Table 50-1](#) plus 4 clock cycles.

The RSTOUT pin can also be asserted by a write to the SER bit of the System Reset Control Register (SIU_SRCR); however, an internal reset cycle is not initiated under this circumstance. The duration of RSTOUT assertion caused by setting the SER bit depends on whether the clock source is the IRC or not, as given in the following table.

1. The FCCU needs to be reset after a self-test run, because FCCU logic is part of the LBIST tests.

Table 50-1. Clock Cycles for Different Reset Sources

Source ¹	Description	Number of clock cycles
PORS	Power-on reset	2400
ERS	External reset (RESET pin)	2410
FOSURS	FOSU reset	2430
FCCURS	FCCU reset	2460
CPURS	Simultaneous core 0 and core 1 reset	2490
STCURS	STCU reset	2500
SSRS	Software system reset	2510
SERF ²	Software external reset	2520 (IRC) / 17200 (other)

1. Bits of the SIU_RSR register.

2. RSTOUT duration depends on the clock source: if IRC, duration is 2520; for all others, duration is 17200.

50.4 Reset sources

50.4.1 Power-on reset

The internal power-on reset signal is asserted when voltage and temperature monitors indicate that the device temperature and/or power supply voltages have reached values outside normal operating ranges. Control and configuration registers for temperature and voltage monitors are located within the Power Management Control (PMC) digital interface module.

POR is the highest priority reset and if POR is detected while any other reset is being processed, after reset cycle completion SRS will have just the PORS set.

50.4.1.1 Voltage monitors

The following out-of-range voltage events can trigger a power-up reset cycle:

- POR: below absolute minimum value required for starting device wake-up process
- Low voltage Detection (LVD): below minimum power supply voltage specification
- High Voltage Detection (HVD): above maximum power supply voltage specification

A power-up reset cycle is triggered unconditionally when supply voltages fall below POR thresholds. LVD and HVD events can also trigger a power-up cycle if enabled in the PMC module. LVD/HVD events are also routed from the PMC to the FCCU to be handled there.

LVD/HVD events which are enabled to trigger reset will cause a POR reset cycle. When the reset cycle finishes, the PORS flag is set in the SIU_RSR register. If LVD/HVD are routed to the FCCU, and the FCCU is enabled to generate reset from these sources, then an FCCU reset cycle is triggered. When the FCCU reset cycle finishes, the FCCURS bit is set in the SIU_RSR register. The FCCU reset cycle will not reset the FCCU itself, therefore information about the LVD/HVD events can still be recovered from the FCCU registers.

See the the PMC chapter and the device data sheet for information on monitored voltages and threshold values.

50.4.1.2 Temperature monitors

The following table identifies the temperatures for which out-of-range temperature events can trigger a power-up reset cycle:

Table 50-2. Temperature events configurable to cause POR

Temperature	Temperature Sensor name	PMC name	Out-of-range event
-40°C	Low temperature	Temperature 0	Temperature is below this threshold
150°C	High temperature 1	Temperature 2	Temperature is above this threshold
165°C	High temperature 2	n/a	No event: This device does not support 165°C monitoring.

These events trigger a power-up cycle if enabled in the PMC module. Alternatively, the PMC can be configured to route the temperature events to the FCCU to be handled there.

Temperature events that are enabled to trigger a reset cause a POR cycle. When the reset cycle finishes, the PORS flag is set in the SIU_RSR register. If the temperature events are routed to the FCCU, and the FCCU is enabled to generate reset from these sources, then an FCCU reset cycle is triggered. When the FCCU reset cycle finishes, the FCCURS bit is set in the SIU_RSR register. The FCCU reset cycle will not reset the FCCU itself, so information about the LVD/HVD events can still be recovered from the FCCU registers.

See the detailed descriptions of the Temperature Sensor and PMC modules as well as the device Data Sheet for information about monitored temperatures and threshold values.

50.4.2 External reset

Assertion of the RESET pin triggers an internal reset cycle. The reset cycle will be sustained as long as RESET is kept asserted. When RESET is negated, the reset controller initiates a timer that counts for a predetermined number of clock cycles, as shown in [Table 50-1](#), before resuming the reset cycle.

If the RESET pin was the source reset, or if it was asserted during the last reset cycle, then the ERS bit is asserted in the SIU_RSR register.

If POR is the current reset source being processed, then a RESET pin is detected, after reset cycle completion SRS will have PORS and ERS set.

If the Reset pin is the current reset source being processed, then if any other reset is detected (except POR), after reset cycle completion SRS will have just ERS set and PORS will keep its previous value.

50.4.3 FCCU reset

The Fault Collection and Control Unit collects failure events coming from different sub-systems within the device. One of the possible reactions to a failure event is the triggering of an internal reset cycle.

Failure sources include:

- All failure indication signals from modules within the MCU
- Logic and signals monitored by the FCCU itself
- Software-initiated failure indications
- External failure input.

Available failure reactions are:

- Assertion of an interrupt (maskable or nonmaskable)
- Resetting the MCU
- Changing the state of the failure indication pins, ERRORn
- No reaction

The FCCU itself is not affected by an FCCU reset. The FCCU is reset when the reset source is one of the following:

- Power-up
- RESET pin
- FOSU reset
- STCU reset

When the reset cycle that was initiated by the FCCU finishes, the FCCURS bit is asserted in the SIU_RSR register. Since the FCCU itself is not reset when the reset source is an FCCU reset, information about the failure that caused reset is still available in the FCCU module to be read by software.

Refer to the FCCU chapter for information on failure sources that can generate a reset cycle.

50.4.4 FOSU reset

As the FCCU is a central component in reacting to errors, it is itself supervised. This supervision is provided by the FCCU Output Supervision Unit (FOSU). The FOSU receives failure indications at the same time as the FCCU. Unless the respective failure is switched off, the FOSU will observe the outputs of the FCCU (IRQ, RESET, error pins). If the FCCU does not react within a predefined timing window, the FOSU assumes the FCCU has failed and causes a reset cycle.

A FOSU reset cycle resets the FCCU and the FOSU itself. The FOSU is reset when the reset source is one of the following:

- Power-up
- RESET pin
- FOSU reset
- STCU reset

When the FOSU reset cycle finishes, the FOSURS bit is asserted in the SIU_RSR register.

50.4.5 Dual core reset

Core 0 and Core 1 Reset Vector Registers described in the SIU chapter allow software executing on either core to place the other core in reset by setting the RST bit in the appropriate Reset Vector Register. Any attempt to reset both cores by setting both RST bits will result in an immediate internal reset.

When the reset cycle that was initiated by this reset source finishes, the CPURS bit is asserted in the SIU_RSR register.

50.4.6 JTAG reset

An internal reset cycle is initiated when JTAG is enabled and either the EXTEST, CLAMP, or HIGHZ instructions are executed by the JTAG controller. Since these instructions affect the state of the package pins, the state of the RSTOUT pin is determined by the JTAG instruction itself. The reset status bits in the SIU_RSR are not affected by this reset source.

50.4.7 Software system reset

A software system reset occurs when software asserts the SSR bit in the System Reset Control Register (SIU_SRCR), triggering an internal reset cycle and the assertion of the RSTOUT pin. Once the reset cycle begins, the SSR bit is automatically cleared. When the reset cycle finishes, the SSRS bit is set in the SIU_RSR register.

50.4.8 Software external reset

A write of one to the SER bit in the SIU_SRCR causes the external RSTOUT pin to be asserted for a predetermined number of clock cycles (refer to [RSTOUT](#)). The SER bit automatically clears after the clock counting expires. A software external reset does not cause an internal reset cycle. Only the RSTOUT pin is affected. The SERF bit in the SIU_RSR is set, but no other status bits are affected. The SERF bit in the SIU_RSR is not automatically cleared after the RSTOUT clock cycles expire, and remains set until cleared by software or another reset besides the software external reset occurs.

During a software external reset, the e200z7 cores will continue to execute instructions, timers that are enabled will continue to operate, and interrupt requests will continue to be processed. It is the responsibility of the application to ensure devices connected to RSTOUT are not accessed during a software external reset, and to determine how to manage MCU resources when using the software external reset.

50.4.9 STCU reset

When the reset cycle is triggered by a power-up event or by the assertion of the RESET pin or a FOSU reset, and if enabled by DCF configuration, the Self-Test Control Unit executes a series of hardware self tests based on shifting pseudo-random data into the device registers and then comparing the resulting signature with a golden reference stored in flash memory. It also triggers the execution of self-test hardware within the embedded

memories. By the end of the self-test runs, the device is left with random data in its registers; therefore, the STCU sends a signal to the reset controller to trigger a new reset cycle.

The STCU itself is not affected by STCU reset cycle; therefore, test results are preserved to be read by software after the reset cycle finishes. The FCCU is also one of the structures included in the STCU tests; therefore, it also needs to be reset after the execution of STCU diagnostics.

When the reset cycle that was initiated by this reset source finishes, the STCURS bit is asserted in the SIU_RSR register.

50.4.10 Life cycle manipulation reset

When SSCM loads information from UTest memory corresponding to the life cycle slots and that information includes invalid programmed values, a life cycle manipulation reset occurs. See [SSCM security reset and LC state](#) for further details.

50.5 Reset configuration

During reset, the SSCM reads various configuration and trimming data from flash memory and applies them to the device before the reset cycle finishes. See the DCF chapter for a complete list of DCF records. Following is a list of main functions configured by DCF records and information in other locations in UTest flash memory:

- Pull up/down of external I/O pins
- Boot mode (internal, serial, EBI, auto-baud)
- Crystal oscillator trimming and configuration
- RC oscillator trimming
- Trimming for voltage regulators and power supply monitors
- ADC trimming
- Temperature sensor trimming
- Flash access control bits and passwords
- Security, life cycle, and censorship
- STCU configuration

50.5.1 Weak pull up/down configuration

Initial pull up/down configuration of external I/O pins is available in the Miscellaneous DCF record, which is read by the SSCM from UTEST flash memory during reset and applied to the pad drivers.

As an alternative to the DCF record method, a legacy mode is available that uses the WKPCFG pin to do the same configuration. For all resets except a software external reset, the WKPCFG pin is applied at the assertion of the internal reset signal and latched upon negation of reset. Legacy mode is used when the legacy bit is asserted in the Miscellaneous DCF record.

Either coming from the DCF record or from the WKPCFG pin, the pull configuration is stored in the WKPCFG bit in the Reset Status Register.

50.5.2 Boot configuration

Configuration data required by the boot code is available in the Miscellaneous DCF record, which is read by the SSCM from UTEST flash memory during the reset cycle. As an alternative to the DCF record method, a legacy mode is available in which the boot configuration data is latched from package pins at the end of the reset cycle. Either coming from the DCF record or from package pins, the boot configuration data is placed in the Reset Status Register (SIU_RSR) to be read by the boot code.

The boot configuration data in SIU_RSR is also affected by the security life cycle of the component. If the life cycle has reached the “in field” state, then boot mode is forced to be internal only. See the table below and also refer to the SIU_RSR description in the SIU chapter.

Table 50-3. Boot configuration data

Mode (LEG bit in Misc DCF record)	SIU_RSR fields	
	BOOTCFG ¹	ABR
Normal (LEG=0)	Loaded from bits 14–15 of Misc DCF	Loaded from bit 13 of Misc DCF
Legacy (LEG=1)	Latched from BOOTCFG[0:1] pins	Latched from EVTO pin ²

1. BOOTCFG is forced to 0b00 (internal boot) if life cycle is “in-field.”

2. Latching from EVTO works as follows. If no debug tool is attached to the device and EVTO is held at low logic level during reset, the ABR bit reset state is logic one. Otherwise it is zero.

50.6 Boot sequence

When the reset cycle finishes, the following sequence of events takes place:

1. In parallel, immediately after reset is negated, the Cryptographic Services Engine (CSE) starts running its secure boot code, and core 0 starts executing boot code from address 0xFFFF_FFFC.
2. Core 1 is held in reset state until software writes a boot vector in SIU_RSTVEC1 and then negates SIU_HLT1[CORE1] or SIU_RSTVEC1[RST1]. When either of these bits is negated, core 1 starts executing code at the address specified by the boot vector.
3. Execution of the checker core is controlled by the LOCKSTEP_EN bit of the Miscellaneous DCF Client. If disabled, the checker core is kept in reset state after the end of the reset cycle even if core 1 starts executing code. When the checker core is disabled, RCCU error checking is also disabled. Enabling or disabling checker core execution requires a reset cycle to take effect.

Chapter 51

System Boot

51.1 Overview

The Boot Assist Module (BAM) is a 4-KB block of read-only memory, containing the boot program code for the MCU. The BAM program supports the following boot modes:

- Boot from internal flash memory
- Serial boot via SCI or CAN interface with optional baud-rate detection (available if the device life cycle is not “in-field”)
- Boot from a memory connected to the MCU development bus (EBI)¹ with multiplexed or separate address and data lines (available if the device life cycle is not “in-field”)

On the MPC5777C device, only Core0 is enabled out of reset; therefore, the BAM program is executed by Core0 after the reset cycle finishes. Depending on the boot mode, the program initializes the appropriate minimum MCU resources to start user application code. Note that the BAM is pure software; therefore, all registers and resources accessed and modified by the BAM can also be accessed by customer software.

51.2 Features

- Initial MMU setup with minimum address translation for all internal MCU resources
- MMU configuration to boot user application, compiled as Classic PowerPC Book E code or as NXP VLE code
- Passes control to user application code in the internal flash memory or external memory device
- Automatic switch to serial boot mode if internal or external flash is blank or invalid
- Serial boot by loading user program via CAN bus or eSCI to the internal SRAM
 - Not available at “in-field” device life cycle

1. EBI not available on all packages.

- Only public password protection is used
- Optional automatic detection of the host SCI or CAN speed
- Boot from an external memory device, connected to the EBI
 - Not available at “in-field” device life cycle
 - Option to boot from 16-bit memory device with separate data and address lines
 - Option to boot from 32/16-bit memory device with multiplexed data and address lines
- Controls MCU core Watchdog Timer and/or the Software Watchdog Timer (SWT)

51.3 Modes of operation

51.3.1 Normal boot

Core 0 is released from reset immediately at the end of the reset cycle and it begins execution of the BAM code from the reset vector contained in the SIU_RSTVEC0 register.

51.3.2 Parallel secure boot

The Cryptographic Service Engine (CSE) and Core0 are released from reset at the same time. Core0 starts executing the BAM code in parallel with CSE execution. The BAM code is ignorant of the result of the secure boot code execution; it always continues with its own code execution.

51.3.3 Debug mode

The BAM program is not executed when the MCU exits reset in OnCE debug mode. Before accessing the MCU resources, use the development tool to initialize the MCU.

51.3.4 Internal boot mode

BAM looks for customer code at predefined locations in flash memory, then starts execution of that code from flash.

51.3.5 Serial boot mode

BAM loads customer code into internal SRAM using the eSCI or CAN serial interface, then executes that program. The program can then control:

- Downloading of data, and
- Erasing and programming of the internal or external flash memory

51.3.6 Development bus boot mode

This boot mode is intended for packaged versions of the device that have the development bus pinned out. The development bus (EBI) boot mode can be used for application code development.

51.4 Memory Map

The BAM occupies 16 KB of memory space, 0xFFFF_C000 to 0xFFFF_FFFF. The actual code size of the BAM program is less than 4 KB and starts at 0xFFFF_F000, repeating itself every 4 KB in the BAM address space. The CPU starts the BAM program execution at its reset vector from address 0xFFFF_FFFC.

The BAM exits to user code at 0xFFFF_FFF8. The last instruction the BAM executes is a BLR. The link register is pre-loaded with the user application start address. The value of the start address depends on the boot mode:

- Booting from internal or external flash memory—32-bit word following a valid RCHW holds the start address value
- Serial boot—set according to the serial boot protocol

The following table shows the BAM address map.

Table 51-1. BAM memory map

Address	Description
0xFFFF_C000 – 0xFFFF_EFFF	BAM program mirrored
0xFFFF_F000 – 0xFFFF_FFFF	BAM program
0xFFFF_FFFC	MCU reset vector
0xFFFF_FFF8	BAM last executed instruction

51.5.1 BAM program flow chart

```

graph TD
    Reset([Reset]) --> ConfigMMU[Config. MMU for internal boot]
    ConfigMMU --> BOOTCFG00{BOOTCFG = 00 or  
Life Cycle = In Field}
    BOOTCFG00 -- Y --> SearchRCHW[Search for RCHW]
    SearchRCHW --> FoundRCHW1{Found RCHW?}
    FoundRCHW1 -- Y --> ParseRCHW1[Parse RCHW and execute RCHW options]
    FoundRCHW1 -- N --> LifeCycle{Life Cycle = In Field?}
    LifeCycle -- Y --> SerialBoot[Serial boot]
    LifeCycle -- N --> BOOTCFG01{BOOTCFG = 01}
    SerialBoot --> ClockStable{Clock Stable?}
    ClockStable -- N --> ClockStable
    ClockStable -- Y --> Exit([Exit To User Code])
    BOOTCFG01 -- Y --> SerialBoot
    BOOTCFG01 -- N --> BOOTCFG10{BOOTCFG = 10}
    BOOTCFG10 -- Y --> SetupEBISeparate[Setup EBI and development bus pins for separate address and 16 bit data operation]
    BOOTCFG10 -- N --> SetupEBIMultiplexed[Setup EBI and development bus pins for multiplexed address and 16 bit data operation]
    SetupEBISeparate --> ConfigMMUEBI[Configure MMU for EBI boot]
    SetupEBIMultiplexed --> ConfigMMUEBI
    ConfigMMUEBI --> CheckRCHW[Check RCHW]
    CheckRCHW --> FoundRCHW2{Found RCHW?}
    FoundRCHW2 -- Y --> ParseRCHW2[Parse RCHW and execute RCHW options]
    FoundRCHW2 -- N --> SerialBoot
    ParseRCHW1 --> Exit
    ParseRCHW2 --> Exit

```

Figure 51-1. BAM program flow chart

51.5.2 BAM program operation

When Core0 is released from reset state, it fetches the boot vector corresponding to the BAM code from the SIU_RSTVEC0 register. The first thing the BAM program does is to configure the MMU to allow access to all MCU internal resources, according to [Table 51-2](#). This MMU setup remains untouched through internal flash boot mode.

Table 51-2. MMU configuration for internal flash boot

TLB entry	Region	Logical base address	Physical base address	Size	Attributes
0	Peripheral Bridge B and BAM	0xFFE0_0000	0xFFE0_0000	2 MB	Guarded
					Big endian
					Global PID
					Cache Inhibit
1	Internal flash	0x0000_0000	0x0000_0000	16 MB	Not guarded
					Big endian
					Global PID
2	EBI	0x2000_0000	0x0000_0000	16 MB	Not guarded
					Big endian
					Global PID
3	Internal SRAM	0x4000_0000	0x4000_0000	512 KB	Not guarded
					Big endian
					Global PID
					Cache Inhibit
4	Peripheral Bridge A	0xC3E0_0000	0xC3E0_0000	2 MB	Guarded
					Big endian
					Global PID
					Cache Inhibit

The MMU regions are mapped with logical addresses the same as physical addresses except for the External Bus Interface (EBI). The logical EBI address space is mapped to physical address space of the internal flash memory. This allows code, written to run from external memory, to execute from internal flash memory.

After the MMU configuration, the BAM uses information from the read-only SIU_RSR[BOOTCFG] field to decide what type of boot that it must do, according to [Table 51-3](#).

Table 51-3. BOOTCFG field description

BOOTCFG[0:1]	Description
00	Internal boot from flash memory
01	Serial boot from eSCI or FlexCAN

Table continues on the next page...

Table 51-3. BOOTCFG field description (continued)

BOOTCFG[0:1]	Description
10	Non-muxed EBI boot (16-bit data, non-muxed address bus)
11	Muxed EBI boot mode (16-bit data, muxed address bus)

The BOOTCFG field is the only place that the BAM will look to determine what type of boot it should attempt. The value of the BOOTCFG field is loaded into the SIU_RSR by internal hardware during reset, from one of the following possible sources:²

1. BOOTCFG pins
2. BOOTCFG field of the Miscellaneous DCF record
3. Fixed value 00 (Internal Boot)

Which of these sources is used depends on the life cycle and the legacy mode settings of the device.

Table 51-4 describes the source selection of the BOOTCFG field depending on the above settings.

Table 51-4. BOOTCFG source selection

Life cycle	Legacy mode?	BOOTCFG source
Different than In Field	Yes	BOOTCFG pins
	No	BOOTCFG field of Miscellaneous DCF record
In Field	X ¹	Fixed to 00 (Internal Boot)

1. "X" means that the value of this entry has no effect on the determination of the BOOTCFG source.

For all cases presented where serial boot mode is available, a 64-bit public password is required. The password is 0xFEED_FACE_CAFE_BEEF. After a match is detected, the BAM code goes on executing the serial boot flow. Otherwise, it stays on a loop until a reset is issued due to the watchdog settings.

The internal and EBI boot modes require no password, but it is still possible to have the flash memory inaccessible due to protection settings on the PASS module. Therefore, access to flash should be granted by not blocking the flash memory regions used by the BAM.

2. The BOOTCFG field is the only place that the BAM will look to determine what type of boot it should attempt. Other device settings affect the boot mode, but they are all factored in by the internal hardware when it loads BOOTCFG into the SIU_RSR register.

51.5.3 Reset Configuration Half Word (RCHW)

The Reset Configuration Half Word defines boot options and must be programmed to predefined locations in the internal flash or at the beginning of the external flash device. The 32-bit word after the RCHW must be programmed with the user application's starting address. The BAM passes control to the user application at this starting address.

Table 51-6 provides possible RCHW locations in the internal flash. When booting from the external flash device, the RCHW must reside in the first 16-bit half word of the flash memory.

BOOT_BLOCK_ADDRESS

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
				SWT	WTE	PS0	VLE	0	1	0	1	1	0	1	0
								Boot Identifier = 0x5A							

Table 51-5. RCHW field descriptions

Field	Description
0–3	Reserved. These bit values are ignored when the halfword is read. Program to 0 for future compatibility.
4 SWT	Software watchdog timer enable. This bit determines if the software watchdog timer is enabled after passing control to the user application code. 0 Disable software watchdog timer 1 Enable software watchdog timer after reset. The time-out period is 392400 cycles of the Internal RC Oscillator.
5 WTE	MCU core watchdog timer enable. This bit determines if the core software watchdog timer is enabled after passing control to the user application code. 0 Disable core software watchdog timer 1 Enable core watchdog timer after reset. The timeout period is 2.5×2^{18} cycles of the CPU clock. ¹
6 PS0	Port size. Defines the width of the data bus connected to the memory on D_CS0. After system reset, the BAM changes D_CS0 to a 16-bit port to fetch the RCHW from either 16- or 32-bit external memories. Then the BAM reconfigures the EBI as a 16- or 32-bit port, depending on this bit. 0 32-bit D_CS0 port size 1 16-bit D_CS0 port size NOTE: Used in development bus boot modes only (that is, 496- and 516-pin packages). Do not clear this bit if the device only has a 16-bit data bus.
7 VLE	VLE Code Indicator. This bit configures the MMU entries 1–3 coded as Classic Book E instructions or as NXP VLE instructions. 0 User code executes as classic Book E code 1 User code executes as NXP VLE code
8–15 BOOTID[0:7]	Boot identifier. This field serves two functions: - Indicates which block in flash memory contains the boot program - Indicates if the flash memory is programmed (BOOTID = 0x5A) or invalid

1. Upon exiting reset, the CPU clock is connected to the Internal RC Oscillator at 16 MHz.

51.5.3.1 Application Start Address Register

Application Start Address Register, shown below, is the 32-bit word after the RCHW in the application code memory device (internal or external flash memory). The BAM uses the value from this location as a start address of the user application to pass control to.

BOOT_BLOCK_ADDRESS + 0x0000_0004

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
A0	A1	A2	A3	A4	A5	A6	A7	A8	A9	A10	A11	A12	A13	A14	A15

16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
A16	A17	A18	A19	A20	A21	A22	A23	A24	A25	A26	A27	A28	A29	A30	A31

51.5.4 Internal boot mode

Upon reset exit, when the BAM program starts running, the system clock is by default sourced by the Internal RC Oscillator (IRC). If internal boot mode was configured in the SIU_RSR[BOOTCFG] field, the BAM program executes the following tasks:

1. Configures a machine check exception handler, since it will be accessing flash memory locations that may be corrupted and cause a bus error
2. Searches for a valid RCHW in predefined locations, as shown in the following table.

NOTE

In the search for a valid RCHW, the location with priority 0 has highest priority and the location with priority 4 has lowest priority.

Table 51-6. Possible RCHW locations in internal flash memory

Priority	Address
0	0x0080_0000
1	0x0002_0000
2	0x0003_0000
3	0x0000_0000
4	0x0001_0000

3. If a valid RCHW is not found, the BAM program checks for the life cycle state of the device (SSCM_LCSTAT[LC]). For any life cycle other than In Field, the BAM program switches to the serial boot mode. Otherwise, the search for the RCHW will continue until SWT requests a reset.
4. If the BAM program finds a valid RCHW:

- `BOOT_BLOCK_ADDRESS` is the address (from [Table 51-6](#)) of the valid RCHW
- The BAM program fetches the application start address from `BOOT_BLOCK_ADDRESS + 0x4`
- The BAM branches to this start address (shown in [Application Start Address Register](#))
- The MMU TLB entries 1,2,3 are programmed with VLE attribute if the `RCHW[VLE] = 1`
- The watchdog timers are programmed as shown in the table below

Table 51-7. Watchdog timeout settings

Core WD		SWT	
Clock source	Condition	Clock source	Condition
IRC 16 MHz	Enabled if <code>RCHW[WTE] = 1</code> Timeout = 2.5×2^{18} cycles	IRC 16 MHz	Enabled if <code>RCHW[SWT] = 1</code> Timeout = 392400 cycles

51.5.5 Serial boot mode

When serial boot mode is detected, either because of the `BOOTCFG` configuration or because a valid RCHW word was not found in any of the expected locations in flash memory, the BAM program does the following:

1. Initializes the FlexCAN_A RAM to avoid ECC errors
2. Uses message buffers in FlexCAN_A for program stack and variables
3. Polls the `SIU_RSR[XOSC]` bit until it indicates that the oscillator has locked
4. Switches the clock source to the crystal oscillator by setting `SIU_SYSDIV[SYSCLKSEL] = 0x1`
5. Disables the core watchdog timer and sets the SWT timeout period to 2.5×2^{27} cycles
6. Waits for the host to send a 64-bit password by either the CAN or SCI
7. Receives start address, size of download code in bytes, and VLE bit
8. Receives the application code data
9. Disables the SWT and enables the core watchdog timer with a timeout period of 2.5×2^{28} CPU clock cycles
10. Branches to the loaded code at the start address

The MMU setup depends on how the BAM entered serial boot mode. If it attempted to boot from the development bus but switched to serial boot because it could not find a valid RCHW in Flash, the MMU is set up as for that mode (see [Table 51-13](#)). Otherwise, the MMU setup matches [Table 51-2](#).

Serial boot mode can run in two modes, depending on the state of the SIU_RSR[ABR] bit:

- Standard serial boot mode—fixed baud rates derived from the MCU system frequency. EVTO pin is kept high during reset.
- Baud Rate Detection serial boot mode—allows communication with adaptable speed, based on measured baud rate of the host transmission. EVTO pin has to be driven low for this mode.

In enhanced mode, the SIU_RSR[ABR] bit is loaded from the ABR bit of the Miscellaneous DCF record. In legacy mode, the latching of the EVTO pin status in the SIU_RSR register is done following ways:

- If no debug tool is attached, then the state of EVTO pin is latched upon negation of system reset
- If a debug tool, using EVTO pin, is attached, the state of EVTO pin is latched upon enabling the Nexus module

The EVTO pin is pulled up by an internal pull-up. Actively driven low EVTO pin selects the Baud Rate Detection mode.

When Fixed Baud Rate mode is selected, the BAM program configures the RXD_A pin to be the input of the eSCI_A module, CNRX_A pin as an input and CNTX_A as an output of the CAN_A module. When Baud Rate Detection Mode is selected, the BAM program configures RXD_A and CNRX_A pins as GPI inputs for polling their state by the CPU.

The SCI and CAN controller pin configuration summaries are shown in [Table 51-8](#).

Table 51-8. CAN/eSCI pin configuration for CAN/eSCI fixed baud rate boot modes

Pins	Reset Function	Initial Serial Boot Mode		Serial Boot Mode after a valid CAN message received		Serial Boot Mode after a valid eSCI message received	
		Function	Pad Configuration	Function	Pad Configuration	Function	Pad Configuration
CNTX_A	GPIO	CNTX_A	Push/Pull output, with medium slew rate	CNTX_A	Push/Pull output, with medium slew rate	GPIO	—
CNRX_A	GPIO	CNRX_A	Input with pullup and hysteresis	CNRX_A	Input with pullup and hysteresis	GPIO	—
TXD_A	GPIO	GPIO	—	GPIO	—	TXD_A	Push/Pull output, with medium slew rate
RXD_A	GPIO	RXD_A	Input with pullup and hysteresis	GPIO	—	RXD_A	Input with pullup and hysteresis

The BAM configures the communication modules for reception with fixed baud rates as shown in [Table 51-9](#) and then waits for data reception. The watchdog timers are configured as shown in [Table 51-10](#).

Table 51-9. Serial boot fixed baud rates

Crystal frequency (MHz)	Peripheral clock frequency (MHz)	Desired SCI baud rate (baud)	Actual SCI baud rate (baud)	CAN baud rate ¹ (baud)
f_{xtal}	$f_{\text{per}} = 0.5 \times f_{\text{xtal}}$	—	$f_{\text{per}} \div 416^2$	$f_{\text{per}} \div 40^3$
8	4	9600	9615	100K
12	6	14400	14423	150K
16	8	19200	19230	200K
20	10	24000	24038	250K
40	20	48000	48076	500K

1. The CAN engine is clocked by the system clock.
2. The transmitter clock frequency, which is denoted by the bit time, is determined by the equation $f_{\text{TCLK}} = f_{\text{sys}} \div (16 \times \text{SBR}) = \text{SCI}_{\text{BaudRate}}$ where SBR is a constant defined on BAM code to provide the correct baud rate.
3. The Baud Rate Division Factor is defined as $\text{BRF} = (\text{PRES DIV} + 1) \times \text{Number of Time Quanta}$, where the number of time quanta is dependent on the SYN_REG, Time Segment 1, and Time Segment 2 constants. See [Figure 51-2](#).

Table 51-10. Watchdog timeout settings for serial boot

Core WD		SWT	
Clock source	Condition	Clock source	Condition
XOSC $\div$ 2 4–20 MHz	Disabled during boot Enabled after boot Timeout = 2.5×2^{28} cycles	IRC 16 MHz	Enabled during boot Disabled after boot Timeout = 2.5×2^{27} cycles

If a message containing 8 bytes with ID 0x11 is received by the CAN controller first, the BAM program:

- Transitions to serial CAN boot mode
- Disables the eSCI
- Configures RXD_A to its reset state
- Transitions to the CAN serial download protocol routine

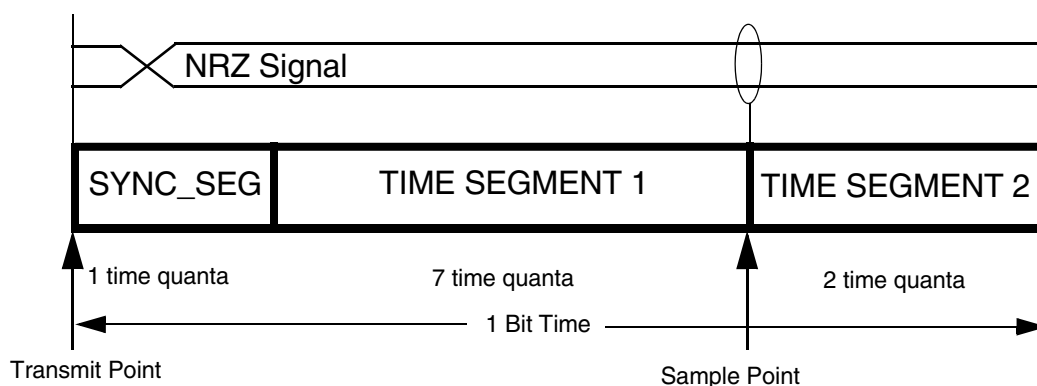
If a byte from eSCI is received first, the BAM program:

- Transitions to the Serial SCI Boot sub-mode
- Disables CAN_A module
- Configures CAN_A's pins to their reset state
- Transitions to the SCI serial download protocol routine

51.5.5.1 CAN controller configuration in fixed baud rate mode

The CAN controller is configured to operate at a baud rate equal to system frequency divided by 40, using the standard 11-bit identifier format detailed in the CAN 2.0A specification. See [Table 51-9](#) for examples of baud rates. Only one message buffer 0 is used for all communications.

The bit timing is configured as shown in the following figure.



Note: 1 Time quanta = 4 System clock periods

Figure 51-2. CAN bit timing

The BAM program ignores CAN errors and all received messages are assumed to be good. Received messages are transmitted back. The host processor must compare the MCU transmissions with the sent data and restart the process if an error is detected.

51.5.5.2 SCI controller configuration in fixed baud rate mode

The eSCI is configured for 1 start bit, 8 data bits, no parity, 1 stop bit and operates at a baud rate equal to system clock divided by 416. See [Table 51-9](#) for examples of baud rates.

The BAM program ignores the eSCI errors. All data received is assumed to be good and is sent back through the TXD pin. The host processor must compare the MCU transmitted data with the sent data and restart the process if an error is detected.

51.5.5.3 Serial boot mode download protocol

The download protocol follows four steps:

1. Host sends 64-bit password.
2. Host sends start address, size of download code in bytes, and VLE bit.
3. Host sends the application code data.

4. The MCU switches to the loaded code at the start address.

The communication is done in half-duplex manner; any transmission from host is followed by the MCU transmission. The host computer should not send data until it receives echo from the MCU. All multi-byte data structures must be sent most-significant byte (MSB) first.

When the CAN is used for serial download, the data is packed into standard CAN messages in the following manner:

- A message with 0x11 ID and 8-byte length to send the password. The MCU transmits back the same data, but with ID of 0x1.
- A message with 0x12 ID and 8-byte length to send the start address, length, and the VLE mode bit. The MCU transmits back the same data, but with ID of 0x2.
- Messages with 0x13 ID are used to send the downloaded data. The MCU transmits back the same data with ID of 0x3.

When the SCI is used for serial download, the data must be sent on a byte-by-byte basis. The MCU transmits back the received bytes.

NOTE

Since the MCU starts up with the system clock derived from the Internal RC Oscillator (IRC), it does not wait for the crystal oscillator to stabilize before finishing the reset sequence and negating the $\overline{\text{RTSTOUT}}$ pin. The BAM program starts executing with the IRC clock, but it needs to switch to the crystal oscillator (XOSC) clock in case of serial boot. Before switching to the XOSC clock, the BAM program waits for the SIU_RSR[XOSC] bit to be asserted, which is an indication that the crystal oscillator has settled. As a consequence, the external device that will upload the serial code needs to wait enough time after $\overline{\text{RSTOUT}}$ negates to allow the crystal oscillator to stabilize. The exact time depends on the crystal used, but is typically less than 10 ms. On the other hand, the code must be uploaded before the SWT timeout expires. The SWT timeout is 21 seconds.

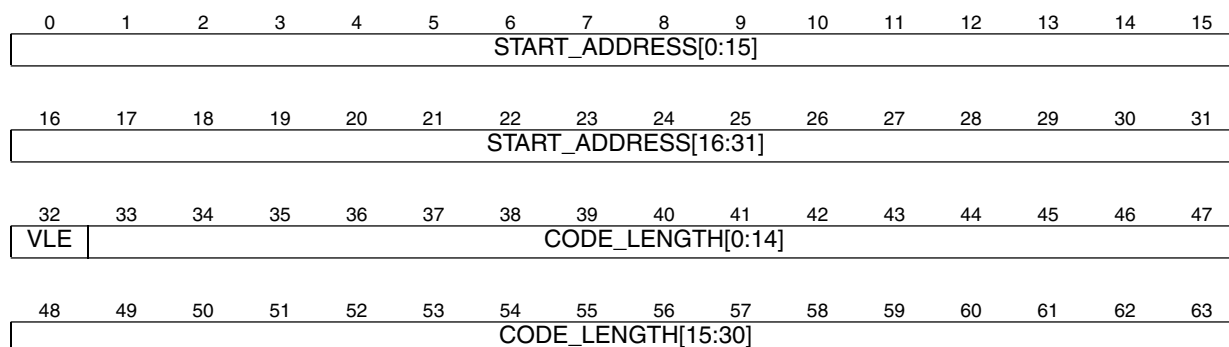
51.5.5.4 Download protocol execution

The BAM program executes the serial boot according to the following steps:

1. Download 64-bit password.
 - The received 8-byte password is checked for validity. For a password to be valid, none of its four 16-bit half words must equal 0x0000 or 0xFFFF.
 - The password is compared to the fixed value of 0xFEED_FACE_CAFE_BEEF.

- If the password check fails, the MCU stops responding. Assert RESET to force the MCU out of this state.
 - If the password check passes, the BAM transitions to the next step in the protocol.
2. Download start address, size of download, and VLE bit.

The BAM considers the next 8 bytes to contain a 32-bit start address, the VLE mode bit, and a 31-bit code length, as shown below.



- START_ADDRESS defines where the received data is stored and where the MCU branches after the download is finished. The two least significant bits of the start address are ignored by the BAM program, thus the loaded code should be 32-bit word aligned.
 - CODE_LENGTH defines how many data bytes to be loaded.
 - VLE instructs the MCU to program MMU entries 1–3 with VLE attribute. If it is 1, the downloaded code must be compiled to VLE instructions, if it is 0 the code contains classic Power Book E architecture instructions.
3. Download data.

Each byte of data received is stored in the MCU memory, starting at the START_ADDRESS specified in the previous step. The data increments through memory until the number of bytes stored matches CODE_LENGTH specified in the previous step.

The BAM program buffers incoming data, collecting up to 8 bytes. The buffered data is written to the RAM with 64-bit writes to prevent ECC errors, which may happen if the MCU RAM is protected by 64-bit ECC code.

Once the buffered data is written to the RAM, the BAM program refreshes the SWT.

NOTE

Only system RAM supports 64-bit writes; therefore, attempting to download data to other RAM apart from system RAM causes errors.

If the start address of the downloaded data is not on an 8-byte boundary, the BAM writes 0x0 to the memory locations from the proceeding 8-byte boundary to the start address (maximum 4 bytes). The BAM also writes 0x0 to all memory locations from the last byte of data downloaded to the following 8 byte boundary (maximum 7 bytes). An additional 8 zero bytes are written to prevent possible ECC errors that may be caused by the CPU pre-fetching.

4. Switch to the loaded code.

The BAM program waits for the last echo message transmission to complete, then the active communication controller is disabled. Its pins revert to GPIO inputs.

To provide compatibility with older devices, the BAM:

- Writes the core's time base registers (TBU and TBL) with 0x0
- Enables the core watchdog to cause a reset after a timeout period of 2.5×2^{28} CPU clock cycles
- Disables software watchdog (SWT)

The BAM code passes control to the loaded code at START_ADDRESS, which was received in step 2 of the protocol.

NOTE

The loaded code must periodically refresh the core watchdog timer or change the timeout period to a value that does not cause resets during normal operation.

51.5.5.5 Baud rate detection procedure

When SIU_RSR[ABR] is set, the BAM program enters into Baud Rate Detection mode.

The following additional steps are taken:

1. The baud rate detection routine is copied to the beginning of the system RAM from the BAM ROM to improve baud rate detection accuracy.
2. The CPU branches to the RAM.
3. The MCU configures the CNRX_A and RXD_A pins as general purpose inputs.
4. The MCU polls these GPIs until one of them goes low.

- If the CNRX_A pin transitions first, the BAM program starts CAN baud rate detection routine, ignoring RXD_A. After detecting the CAN baud rate, the BAM program transitions to the CAN download protocol routine described above.
- If the RXD_A pin transitions first, the SCI baud rate detection and download protocol routines are called, ignoring any further CAN pins activity.

The host, executing the serial boot, should send a test frame so that the MCU can detect the host communication speed. When booting through the SCI, the host should send a zero byte as the test frame. When booting through the CAN, the host should send zero ID, zero length message as the test frame.

51.5.5.5.1 SCI baud rate detection

The host must send a zero byte to allow the MCU to detect the serial link baud rate. The host transmits one start bit, eight zero data bits, and one stop bit (the MCU does not echo it).

The MCU polls the RXD_A pin for high-to-low transition and starts the e200 core Time Base counter (TB). Then the MCU polls for low-to-high transition on the pin and when it happens, the MCU turns off the TB counter. The TB content is used to calculate incoming signal baud-rate. The SCI baud rate is equal to the TB content divided by 288. (Measured over 9 bits with 16 system clocks per bit and the core frequency is two times higher than system frequency).

51.5.5.6 CAN baud rate detection

The host transmits a zero length message with zero 11 bit ID and MCU measures time over 40 bits, polling CNRX_A pin for high and low, according to the sent data. The MCU does not acknowledge this message.

The CAN baud rate depends on the number of quantas per bit and serial clock frequency, which is defined by a prescaler. The CAN baud rate detection routine selects these parameters to maximize number of quantas per bit and achieve minimum difference between measured value and duration of the 40 CAN bits, to be programmed with selected pair of the parameters.

The CAN controller can be programmed with 8 to 25 number of quantas per bit. The bit timing parameters, which can be selected by the baud rate detection routine, are shown in the table below.

Table 51-11. Lookup table for CAN bit timing

Time quanta per bit	Time segment 1		Time Segment 2	RJW
	PROPSEG	PSEG1	PSEG2	
8	1	3	3	2
9	2	3	3	2
10	3	3	3	2
11	4	3	3	2
12	3	4	4	2
13	4	4	4	3
14	5	4	4	3
15	6	4	4	4
16	7	4	4	3
17	8	4	4	3
18	7	5	5	4
19	8	5	5	4
20	7	6	6	4
21	8	6	6	4
22	7	6	6	4
23	8	6	6	4
24	7	7	7	4
25	8	7	7	4

Maximum and minimum speeds of the serial communication modules are defined by the MCU system frequency and shown in the table below.

Table 51-12. Maximum and minimum detectable baud rates (crystal oscillator as reference clock source)

F_{xtal} (MHz)	$F_{sys} = 0.5 \times F_{xtal}$ (MHz)	Max baud rate for CAN ¹ (bps)	Min CAN baud rate ($F_{sys}/25/256$) (bps)	Max baud rate for SCI ($F_{sys}/160$) (bps)	Min baud rate for SCI ($F_{sys}/160/2^{13}$) (bps)
8	4	1M	625	25K	3.0
12	6		937.5	37.5K	4.6
16	8		1250	50K	6.1
20	10		1562.5	62.5K	7.6
40	20		3125	125K	15.2

1. Limited by 1 Mbps by CAN standard.

51.5.6 Booting from the development bus

If the MCU boots in one of the Development Bus boot modes, the BAM program:

- Reprograms the MMU entries for EBI and internal flash (see [Table 51-13](#))
- Sets up the EBI and development bus pins
- Tries to read RCHW from logical address 0x2000_0000

If the valid RCHW is read from that address, the BAM program:

- Reads the user application code start address from 0x2000_0004 address
- Parses RCHW
- Sets up watchdogs
- Updates EBI, SRAM, and internal flash MMU entries (1–3), according to RCHW[VLE]
- Passes control to the user code

If no valid RCHW was read, BAM switches to serial boot mode.

Table 51-13. MMU configuration for development bus boot modes

TBL entry	Region	Logical base address	Physical base address	Size	Attributes
1	Internal flash	0x0000_0000	0x2000_0000	16 MB	Not guarded
					Big endian
					Global PID
2	EBI	0x2000_0000	0x2000_0000	16 MB	Not guarded
					Big endian
					Global PID

51.5.6.1 EBI configuration for separate address and data development bus boot mode

The BAM program sets up EBI related registers as shown in the table below.

Table 51-14. Development bus/EBI register settings for separate address data lines mode

Register	Function	Value	Comments
EBI_CAL_BR0	—	0x2000_0803	Sets the base address to 0x2000_0000, 16-bit wide bus, burst Inhibit
EBI_CAL_OR0	—	0xFF80_00F0	Set 15 wait states, 8MB

Table continues on the next page...

Table 51-14. Development bus/EBI register settings for separate address data lines mode (continued)

Register	Function	Value	Comments
SIU_PCR259–SIU_PCR293	D_ADD[12:30], D_ADD_DAT[0:15]	0x0440	Set pads to 20pf drive strength
SIU_PCR305–SIU_PCR307	D_ADD[9:11]		
SIU_PCR256	D_CS0	0x443	Set pads to 20pf drive strength and pull-up enable
SIU_PCR295	D_WE0		
SIU_PCR297	D_OE		
SIU_PCR298	D_TS		

RCHW[PS0] must be programmed to 1, since the development bus does not support 32-bit port size in that sub-mode.

51.5.6.2 EBI configuration for multiplexed address and data development bus boot mode

The BAM program sets up EBI related registers as shown in the table below.

Table 51-15. Development bus/EBI register settings multiplexed mode

Register	Function	Value	Comments
The following registers are programmed when RCHW[PS0] = 1			
EBI_MCR	—	0x0000_0806	AD multiplexed mode, small port size use D16–31
EBI_CAL_BR0	—	0x2000_0883	Sets the base address to 0x2000_0000,16-bit wide bus, AD multiplexed mode, burst Inhibit
EBI_CAL_OR0	—	0xFF80_00F0	Set 15 wait states, 8 MB
SIU_PCR259–SIU_PCR262	D_ADD[12:15],	0x440	Set pads to 20 pf drive strength
SIU_PCR305–SIU_PCR307	D_ADD[9:11]		
SIU_PCR263–SIU_PCR277	D_ADD_DAT[16:30]	0x840	Set pads to 20 pf drive strength
SIU_PCR257	D_ADD_DAT[31]		
SIU_PCR278–SIU_PCR293	D_ADD_DAT[0:15]	0x440	Set pads to 20 pf drive strength
SIU_PCR256	D_CS0	0x443	Set pads to 20 pf drive strength and pullup enable
SIU_PCR295	D_WE0		
SIU_PCR297	D_OE		
SIU_PCR298	D_TS		
SIU_PCR299	D_ALE		
The following register is reprogrammed when RCHW[PS0] = 0			
EBI_CAL_BR0	—	0x2000_0083	Sets the base address to 0x2000_0000,32-bit wide bus, AD multiplexed mode, burst Inhibit

If the boot memory device is 32 bits wide, the RCHW must be programmed in two first half-words of the memory because when the MCU tries to read RCHW, the EBI is configured to use data lines 16–31 for 16-bit port accesses.

Chapter 52

System Status and Configuration Module (SSCM)

52.1 Chip-specific SSCM information

This chip has one instance of the SSCM. The SSCM is part of the reset and boot subsystem.

52.1.1 SSCM and UTEST flash memory

The SSCM loads data stored in multiple locations of the UTEST portion of flash memory.

52.1.1.1 Life Cycle (LC) slots

To determine the chip's Life Cycle status, the SSCM reads and interprets the LC slots according to the [Life Cycle](#) information. The SSCM_LCSTAT[LC] field presents the results of the determination. The following table shows the LC slots' locations in UTEST flash memory.

Table 52-1. Life Cycle slots in UTEST flash memory

Start address	End address	Bytes	Description
0040_0200h	0040_020Fh	16	Life Cycle slot 0 - MCU Production
0040_0210h	0040_021Fh	16	Life Cycle slot 1 - Customer Delivery
0040_0220h	0040_022Fh	16	Life Cycle slot 2 - OEM Production
0040_0230h	0040_023Fh	16	Life Cycle slot 3 - In Field
0040_0240h	0040_024Fh	16	Life Cycle slot 4 - Failure Analysis

52.1.1.2 System configuration data in DCF records and UTEST_OFFSET location

Some DCF records in UTEST flash memory store data for system configuration. The SSCM scans these DCF records to provide information about the current state and configuration of the system.

On this chip, the UTEST_OFFSET location at which the SSCM looks for DCF records is 0040_0300h.

The following table summarizes these records. See the DCF Records chapter for additional details.

Table 52-2. DCF records in UTEST flash memory scanned by SSCM

Address	Purpose
0040_0300h	STCU2 configuration
	PASS configuration
	Tamper Detect configuration
	Miscellaneous configurations, such as boot mode

52.1.1.3 Sources of errors flagged by SSCM_STATUS[CER]

When set to 1, the CER field in the SSCM System Status (SSCM_STATUS) register indicates that the SSCM has detected a configuration error during reset while loading initial system configuration.

On this chip, this field is set to 1 in these cases:

- When the process of loading any STCU2 DCF record encounters one of the conditions that generates a transfer error listed in the STCU2's IPS bus interface description
- When a DCF record that has the parity attribute is loaded without parity alignment

For a list of DCF clients and the corresponding DCF records on this chip, see Table 5-6.

52.1.2 SSCM security reset and LC state

The SSCM provides a security mechanism to prevent device manipulation while loading the configuration data from dedicated slots in UTEST flash memory.

The SSCM ensures that the Life Cycle slots are correctly programmed and that at least one LC slot is valid. The Life Cycle state can only progress forward. Any attempt to move the LC state backward causes a security reset. This reset occurs in the cases where "System Reset" is the result of the scenarios in [Table 52-14](#).

52.1.3 SSCM_ERROR[PAE] and abort responses to illegal accesses

The SSCM_ERROR register's PAE field controls the abort responses to some illegal accesses to off-platform peripherals.

NOTE

This field has no effect on on-platform peripherals or on peripheral slots that are not allocated.

PAE, when set to 1, generates a transfer error in either of these cases of illegal accesses:

- The CPU tries to access any non-existing peripheral.
- The global module enable memory space is accessed.

Regardless of the PAE field, these cases always generate a transfer error:

- An access is attempted to a halted off-platform module.
- An accessed off-platform peripheral reports a transfer error.

52.2 Introduction

52.2.1 Overview

The System Status and Configuration Module (SSCM) is pictured in the figure below.

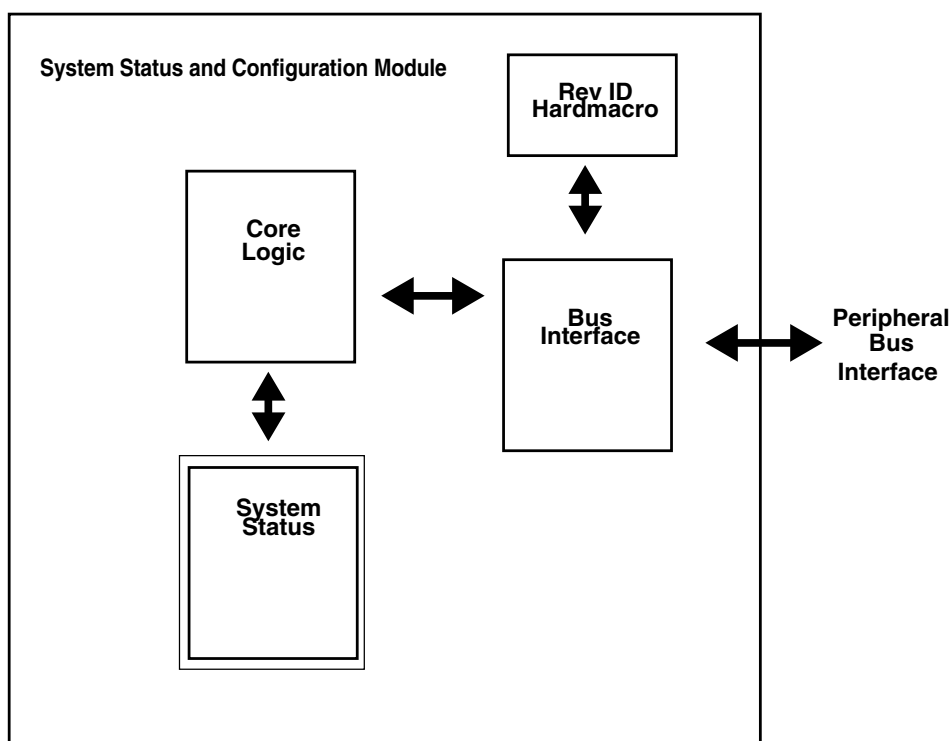


Figure 52-1. SSCM block diagram

52.2.2 Glossary

The table below shows a glossary of terms used through out the text.

Table 52-3. Glossary

Term	Definition
0011b	Binary numbers
0Fh	Hexadecimal numbers
Pin	External physical connection.
Signal	Electronic construct whose state or change in state conveys information.
X	In certain contexts, such as a signal encoding, this indicates a don't care. For example, if a field is binary coded X001b, the state of the first bit is a don't care.
y	y is used for a place holder of 1 hex digit for unknown values.

52.2.3 Features

The SSCM includes these distinct features:

- System Configuration and Status

- Device Mode and System Status
- Decodes the Device Life Cycle
- Peripheral abort enable/disable

52.2.4 Modes of operation

The SSCM operates identically in all system modes.

52.3 External signal description

The SSCM has no external pins.

52.4 Memory map and register definition

This section provides a detailed description of all memory-mapped registers in the SSCM.

The table below shows the memory map for the SSCM. Note that all addresses are offsets; the absolute address may be calculated by adding the specified offset to the base address of the SSCM.

NOTE

All registers are accessible via 8-bit, 16-bit or 32-bit reads and/or writes. However, 16-bit accesses must be aligned to 16-bit boundaries, and 32-bit accesses must be aligned to 32-bit boundaries. As an example, the STATUS register is accessible by a 16-bit read or write, SSCM Base + 02h, but performing a 16-bit access to SSCM Base + 0003h is illegal (that is, you should not make any transfer that is not aligned to burst size boundary).

NOTE

SSCM does not generate transfer errors in all memory holes.

NOTE

Reserved IPS register access is prohibited.

SSCM memory map

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
0	SSCM System Status (SSCM_STATUS)	16	R	See section	52.4.1/2776
6	SSCM Error Configuration Register (SSCM_ERROR)	16	R/W	0000h	52.4.2/2778
34	Life Cycle Status Register (SSCM_LCSTAT)	32	R	See section	52.4.3/2779

52.4.1 SSCM System Status (SSCM_STATUS)

The System Status register reflects the current state of the system.

Address: 0h base + 0h offset = 0h

Bit	0	1	2	3	4	5	6	7
Read	0	CER	MER	NXEN1	NXEN	1	Reserved	0
Write		w1c	w1c					
Reset	0	0	0	*	*	1	*	0
Bit	8	9	10	11	12	13	14	15
Read	Reserved			1	0	0	0	
Write								
Reset	*	*	*	1	0	0	0	0

* Notes:

- Reserved field: Each bit of this field can reset to 0 or 1.
- Reserved field: This field can reset to 0 or 1.
- NXEN field: Reset value depends on the device status after leaving reset.
- NXEN1 field: Reset value depends on the device status after leaving reset..

SSCM_STATUS field descriptions

Field	Description
0 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
1 CER	Configuration Error This field indicates that the SSCM has detected a configuration error during reset while loading initial device configuration. See the chip-specific SSCM information for details about sources of the error. If a non-permanent error is detected, this bit can be cleared by writing a 1. 0 No configuration problem detected by the SSCM 1 Device configuration is not correct
2 MER	Memory Configuration Error This field indicates that the SSCM has received notification of a memory configuration error. This error may have been detected during the reset sequence.

Table continues on the next page...

SSCM_STATUS field descriptions (continued)

Field	Description
	If a detected error is not permanent, this field can be cleared by writing 1 to it. 0 No memory configuration problem detected by the SSCM 1 Device memory configuration is not correct
3 NXEN1	Processor 1 Nexus enable status 0 Processor 1 Nexus disabled. 1 Processor 1 Nexus enabled
4 NXEN	Processor 0 Nexus enable status 0 Processor 0 Nexus disabled. 1 Processor 0 Nexus enabled
5 Reserved	This field is reserved. This read-only field is reserved and always has the value 1.
6 Reserved	This field is reserved. This field is for internal use. Users should ignore read values and should not attempt to write the field.
7 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
8–10 Reserved	This field is reserved. This field is for internal use. Users should ignore read values and should not attempt to write the field.
11 Reserved	This field is reserved. This read-only field is reserved and always has the value 1.
12 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
13 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
14–15 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

52.4.2 SSCM Error Configuration Register (SSCM_ERROR)

The Error Configuration register is a read-write register that controls the error handling of the system.

Address: 0h base + 6h offset = 6h

Bit	0	1	2	3	4	5	6	7
Read	0							
Write								
Reset	0	0	0	0	0	0	0	0
Bit	8	9	10	11	12	13	14	15
Read	0						PAE	Reserved
Write								
Reset	0	0	0	0	0	0	0	0

SSCM_ERROR field descriptions

Field	Description
0–13 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
14 PAE	Peripheral Bus Abort Enable. This bit enables bus aborts on: - any access to a peripheral slot that is not used on the device. - This feature is intended to aid in debugging when developing application code. - illegal accesses to off-platform peripherals. - On platform peripherals cannot be controlled by SSCM peripheral abort. 0 Illegal accesses to non-existing peripherals do not produce a Prefetch or Data Abort exception 1 Illegal accesses to non-existing peripherals produce a Prefetch or Data Abort exception
15 Reserved	This field is reserved.

52.4.3 Life Cycle Status Register (SSCM_LCSTAT)

Address: 0h base + 34h offset = 34h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0						Reserved		0						LC	
W																
Reset	0	0	0	0	0	0	*	*	0	0	0	0	0	*	*	*

* Notes:

- LC field: Reset value is the Life Cycle of the device, and is unaffected by reset.
- Reserved field: Reset Value is '11' for a valid secured part.

SSCM_LCSTAT field descriptions

Field	Description
0–21 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
22–23 Reserved	This field is reserved.
24–28 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
29–31 LC	Life Cycle Determined by the SSCM by reading the LC slots. The encoding is: 000 Failure Analysis 001 Reserved 010 OEM Production 011 Customer Delivery 100 Reserved 101 Reserved 110 MCU Production 111 In Field

52.5 Functional description

The primary purpose of the SSCM is to provide information about the current state and configuration of the system that may be useful for configuring application software and for debug of the system.

52.5.1 DCF mechanism

The DCF mechanism has been developed to handle settings of device parameters via OTP flash memory.

Starting at location UTEST_OFFSET the user can store a series of DCF records - the device will process these records during the system reset sequence before the CPU leaves reset. The first record needs to be a start record followed by an application-specific number of data records as required, and finally a stop record.

Each record is 64 bits in length. [Table 52-8](#) shows the DCF start record. The start record must be placed at UTEST_OFFSET to indicate to the device that the following data records should be processed.

Table 52-8. DCF start record

0:31
05AA 55AFh
0:31
0000 0000h

The data records are structured similar to a CPU write instruction - 15 bits of client select, 15 bits of address, 32 bits of payload data plus a STOP bit - See [Table 52-9](#). In a data record the STOP bit must always be programmed to 0. Some clients support a parity bit.

Table 52-9. Format of a DCF data record

0:31			
WDATA[31:0]			
0:14	15:29	30	31
CS[14:0]	ADDR[16:2]	PRTY	STOP

The stop record indicates that processing should stop. The general format of the stop record is shown in [Table 52-10](#). Only the STOP bit needs to be 1 in order to form a stop record – All other bits are ignored. An unprogrammed location in UTEST flash will be interpreted as a stop record.

Table 52-10. DCF stop record

0:30	31
rsvd	1

Table continues on the next page...

Table 52-10. DCF stop record (continued)

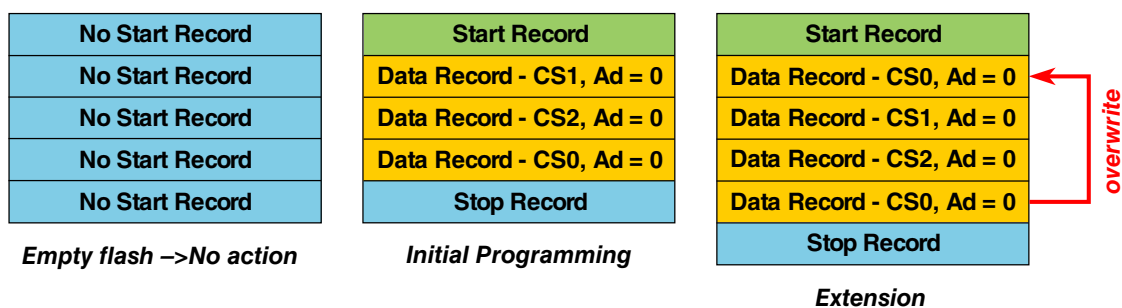
0:31
rsvd

If n data records are to be stored in UTEST, the data structure must be as shown in [Table 52-11](#).

Table 52-11. Series of DCF records in UTEST

ADDR offset	DATA			
00h	05AA 55AFh			
04h	0000 0000h			STOP=0
08h	WDATA[31:0]			
0Ch	CS[14:0]	ADDR[16:2]	PRTY	STOP=0
10h	WDATA[31:0]			
14h	CS[14:0]	ADDR[16:2]	PRTY	STOP=0
...	...			
$8n - 1 + 0h$	reserved			
$8n - 1 + 4h$	reserved			1
$8n + 0h$				
$8n + 4h$				

There must never be an unprogrammed record in the data structure, as that would be interpreted as a stop record, so subsequent records would be ignored. This allows programming the records in several sessions, each time appending new records at the end of the list, as shown in [Figure 52-5](#).

**Figure 52-5. Appending DCF records**

In this case, a record may overwrite a client's value which was already set by a previous record. However whether such an operation is allowed depends on the rules given for the client - some clients are write-once, only allow setting bits but not clearing, and so on.

The parity bit (PRTY) is required for some DCF records. The parity scheme used is even parity, so the number of 1s in the WDATA and PRTY fields needs to be even. (For example, if the WDATA field has the value 0000_0001h then the PRTY field needs to be set to 1, so that the total number of 1s is even.)

NOTE

When programming DCF records, ensure that programming is not interrupted and can complete without error.

52.5.2 Life Cycle

The SSCM will determine the Life Cycle (LC) of the device by reading the LC slots from the UTEST flash memory. The read operation is done during the reset phase with normal timings and it is protected by operating monitors. In addition, a set of sanity checks executed over the LC read data guarantee the integrity of the final LC value.

At the end of the reset phase, the LC can have one of the following values:

- MCU Production
- Customer Delivery
- OEM Production
- In Field
- Failure Analysis

Life cycle can only be moved in the direction shown below:



Figure 52-6. Life cycle sequence

The LC is written into 5 slots, 128 bits each, at fixed locations in the UTEST flash memory block. Each LC slot is read in one single atomic operation and it is organized into two fields:

- the valid field
- the invalid field

Depending on the possible combination of the data programmed into these fields, each LC slot can have one of the possible 4 statuses as seen in [Table 52-12](#).

Table 52-12. LC slot status

LC slots		LC slot value
Valid field (64 bits)	Invalid field (64 bits)	
Erased	Erased	Erased
Marked	Erased	Active
Marked	Marked	Inactive
Any other values		Illegal

"Marked" in this case means that the value has been programmed with the bit pattern 55AA_50AF_55AA_50AFh. "Erased" is detected by the bit pattern FFFF_FFFF_FFFF_FFFFh. [Table 52-13](#) shows how the slots are arranged in memory.

Table 52-13. LC slots in memory

Offset	LC slot word
00h	Valid[31:0]
04h	Valid[63:32]
08h	Invalid[31:0]
0Ch	Invalid[63:32]

The Life Cycle is determined as shown in [Table 52-14](#)

NOTE

"Erased" means the slot returns all 1s.

Table 52-14. LC slots

LC slot 0 MCU Production	LC slot 1 Customer Delivery	LC slot 2 OEM Production	LC slot 3 In Field	LC slot 4 Failure Analysis	Resulting life cycle
Active	Erased	Erased	Erased	Erased	MCU Production
Inactive	Active	Erased	Erased	Erased	Customer Delivery
Inactive	Inactive	Active	Erased	Erased	OEM Production
Inactive	Inactive	Inactive	Active	Erased	In Field
Inactive	Inactive	Inactive	Inactive	Active	Failure Analysis
Erased	Erased	Erased	Erased	Erased	System Reset
Illegal ¹	Illegal	Illegal	Illegal	Erased	In Field

1. Any value programmed other than shown in table above for Erased, Active, and Inactive would be treated as illegal.

NOTE

When programming Life Cycle, ensure that programming is not interrupted and can complete without error.

NOTE

Any other combination of values will result in a system reset.

NOTE

No one should program “ILLEGAL” even if able to do that.
Device behavior is not guaranteed for these combinations.

52.6 Initialization and application information

52.6.1 Reset

The reset state of each individual bit appears in the detailed register descriptions.

52.7 Additional safety measures

52.7.1 Spurious reset protection

The SSCM implements protection against spurious module resets (for example, resets which only effect the SSCM, but not other modules) by gating the state machines status with the expected status of the system reset controller logic. If a spurious reset occurs, the SSCM will not interfere with the flash memory bus or overwrite configuration registers. However, internal SSCM status will be lost.

Chapter 53

Power Management

53.1 Overview

The MPC5777C microcontroller includes a robust power management infrastructure that enables applications to select among various operational modes and to monitor internal voltages for high- and low-voltage conditions. The monitoring capability is also used to ensure supply voltages and internal voltages are within the required ranges before the microcontroller can leave RESET.

The power management infrastructure consists of a Power Management Unit (PMU), which itself consists of a digital portion and an analog portion of a Power Management Controller (PMC).

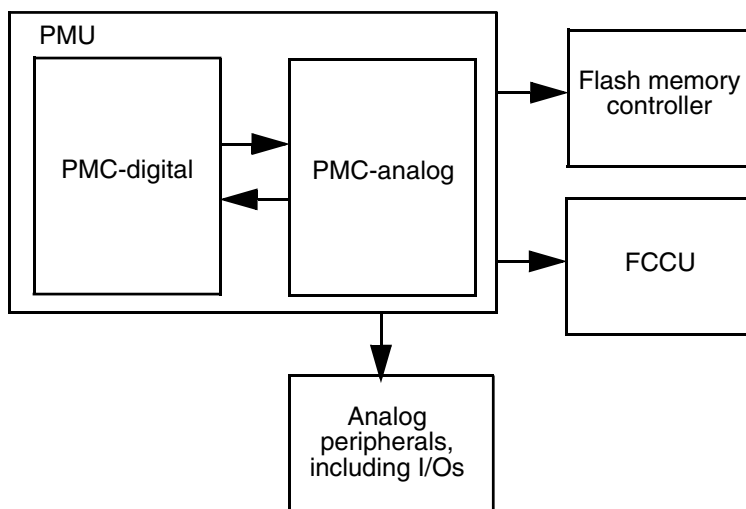


Figure 53-1. Power management overview

For details about any module, see the chapter dedicated to the module.

53.2 Functional description

The analog portion of the PMC provides a linear voltage regulator (LDO) and a switching mode regulator (SMPS) to generate the internal digital core supply and monitors various internal regulating points and supply voltages of the chip for functional safety purposes. The analog portion of the PMC includes an additional regulator for the flash memory block to provide the necessary 3.3 V source to the flash memory module.

The digital portion of the PMC contains the registers and digital logic to generate a power-on reset (POR) in response to particular voltage detection (VD) events and to execute a self test of the analog block. The PMC digital logic also contains the controls for PMC analog outputs to be sensed with the ADC module. In addition, the PMC digital logic contains the enable signals and other register controls for the Temperature Sensor modules.

The PMC connects to the Fault Collection and Control Unit (FCCU) and the flash memory controller. The on-chip ADCs receive signals from the PMC for diagnostic purposes.

53.3 Power modes

MPC5777C does not support ultra-low power modes that require power gating or separate power domains. The core logic implements stop and low frequency power saving modes, which may result in core current steps. To avoid undershoots or overshoots that trigger LVD or HVD events, ramp frequency changes smoothly.

53.4 External supply

The chip provides a bypass option that allows the core supply and flash memory supply to be supplied from an external source.

53.5 Power management supply description

See the product Data Sheet for supply descriptions.

53.6 Voltage monitoring

The PMC senses functional reset and supply voltages continuously to keep the chip in a safe operating range. When the voltage exceeds or falls below the limits defined by recommended electrical operating conditions, the PMC triggers a reset or, alternatively, an interrupt. See the PMC_REE and PMC_LVD_HVD_INTR_ENB registers' descriptions for more information.

Chapter 54

Power Management Controller (PMC)

54.1 Introduction

The Power Management Controller (PMC) consists of two blocks: an analog power management block and a digital controller for the analog block.

The PMC digital controller contains the registers and digital logic to generate a POR in response to voltage-threshold events detected by the analog block. The PMC digital logic also contains:

- the controls for PMC analog outputs to be sensed with the ADC module.
- the enable signals and other registers for the Temperature Sensors.

NOTE

Temperature Sensor 0 and Temperature Sensor 1 each monitor two temperatures: temperature 0 refers to -40°C , and temperature 2 refers to 150°C . Register field notation such as TEMP1_0 designates Temperature Sensor 1's monitoring of temperature 0, or -40°C .

54.1.1 Features

The PMC module provides:

- 16 MHz clock frequency
- 32-bit IPS bus interface
- Memory-mapped configuration, control, and status registers
- Built-in self test for LVD and HVD during non-functional User mode

- Automatic loading of factory-defined trim/control information during POR for HVDs, LVDs, regulators, and voltage bandgap (via the DCF interface)
- Automatic loading of factory-defined trim configuration and control information to the analog PMC block
- Configuration and monitoring of the two Temperature Sensors
- Interrupt or POR generation for HVDs and LVDs (based on the configuration)
 - Option to generate interrupt, POR, or no action in response to:
 - LVD_CORE_COLD
 - HVD_CORE
 - HVD_HV
 - HVD_FLASH
 - Automatic POR generation (except during PMC self test) in response to:
 - POR098_c
 - LVD_CORE_HOT
 - POR_HV
 - LVD_HV
 - LVD_FLASH
 - LVD_IO
- Interrupt or POR generation for Temperature Sensors (based on the configuration)
- Fault injection to the FCCU for LVDs, HVDs, Temperature Sensors, and self test via IPS interface
- Precision LVD (low-voltage detection) and HVD (high-voltage detection)
- Bandgap voltage reference for:
 - LVDs and HVDs
 - ADC voltage reference channel
- FCCU: Software-triggered fault injection for LVDs, HVDs, Temperature Sensors, and LVD self test
- Self test logic for the LVDs and HVDs

54.2 Memory Map and Registers

Attempts to access 32-bit spaces starting at offset D4h result in a transfer error.

For *undocumented* 32-bit spaces at offsets lower than D4h:

- The reset value of the 32-bit space at offset B0h is 0000_0080h.
- The reset value of every other 32-bit space is 0000_0000h.
- Reads of and writes to these locations do not generate a transfer error, but the locations are reserved. Any writes to these locations must write the reset value.

PMC memory map

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
88	Temperature Sensor Interrupt Enable Register (PMC_TS_IER)	32	R/W	0000_0000h	54.2.1/2792
8C	Temperature Event Status Register (PMC_ESR_TD)	32	R/W	0000_0000h	54.2.2/2794
90	Temperature Reset Event Enable register (PMC_REE_TD)	32	R/W	0000_003Fh	54.2.3/2796
94	Temperature Reset Event Selection Register (PMC_RES_TD)	32	R/W	0000_0000h	54.2.4/2798
98	Temperature Detector Configuration Register (PMC_CTL_TD)	32	R/W	0000_0303h	54.2.5/2799
A4	Temperature Sensor Fault Injection Register (PMC_TS_FIR)	32	W	0000_0000h	54.2.6/2800
B4	Voltage Detect User Mode Test Register (PMC_SELF_TEST_UM_VD_REG)	32	R/W	0003_0000h	54.2.7/2801
B8	FCCU Fault Injection Register (PMC_FIR)	32	W	0000_0000h	54.2.8/2804
BC	LVD/HVD Interrupt Enable Control Register (PMC_LVD_HVD_INTR_ENB)	32	R/W	0000_0000h	54.2.9/2805
C0	LVD/HVD Event Status Register (PMC_LVD_HVD_EVENT_STATUS)	32	w1c	0000_0000h	54.2.10/2808
C4	ADC Channel Select Register (PMC_ADC_CS)	32	R/W	0000_0000h	54.2.11/2810
D0	Reset Event Enable Register (PMC_REE)	32	R/W	0000_00FFh	54.2.12/2812

54.2.1 Temperature Sensor Interrupt Enable Register (PMC_TS_IER)

This configuration register contains a set of Interrupt Enable bits, each of which corresponds to a bit in the Temperature Event Status Register (ESR_TD). These bits indicate whether an interrupt occurs when the temperature event occurs. A 0 indicates that no interrupt is to be sent, and a 1 indicates that an interrupt is to occur when the detected temperature event occurs. Enable these bits by writing 1 to the TS_EN field of this register. After TS_EN is 1, this register's Interrupt Enable bits can be read or written at any time.

Address: 0h base + 88h offset = 88h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0								Reserved		TS1_2IE	TS1_0IE	Reserved	TS0_2IE	TS0_0IE	
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

PMC_TS_IER field descriptions

Field	Description
0 TS_EN	Temperature Sensor interrupts Enable 0 Temperature sensor interrupts are not enabled. 1 Enable temperature sensor interrupts.
1–25 Reserved	Reserved This field is reserved. This read-only field is reserved and always has the value 0.
26 Reserved	Reserved This field is reserved.
27 TS1_2IE	Temperature Sensor 1 input 2 Interrupt Enable This bit determines whether an interrupt is seen by the system when the temperature rises above its corresponding threshold (flag 2).

Table continues on the next page...

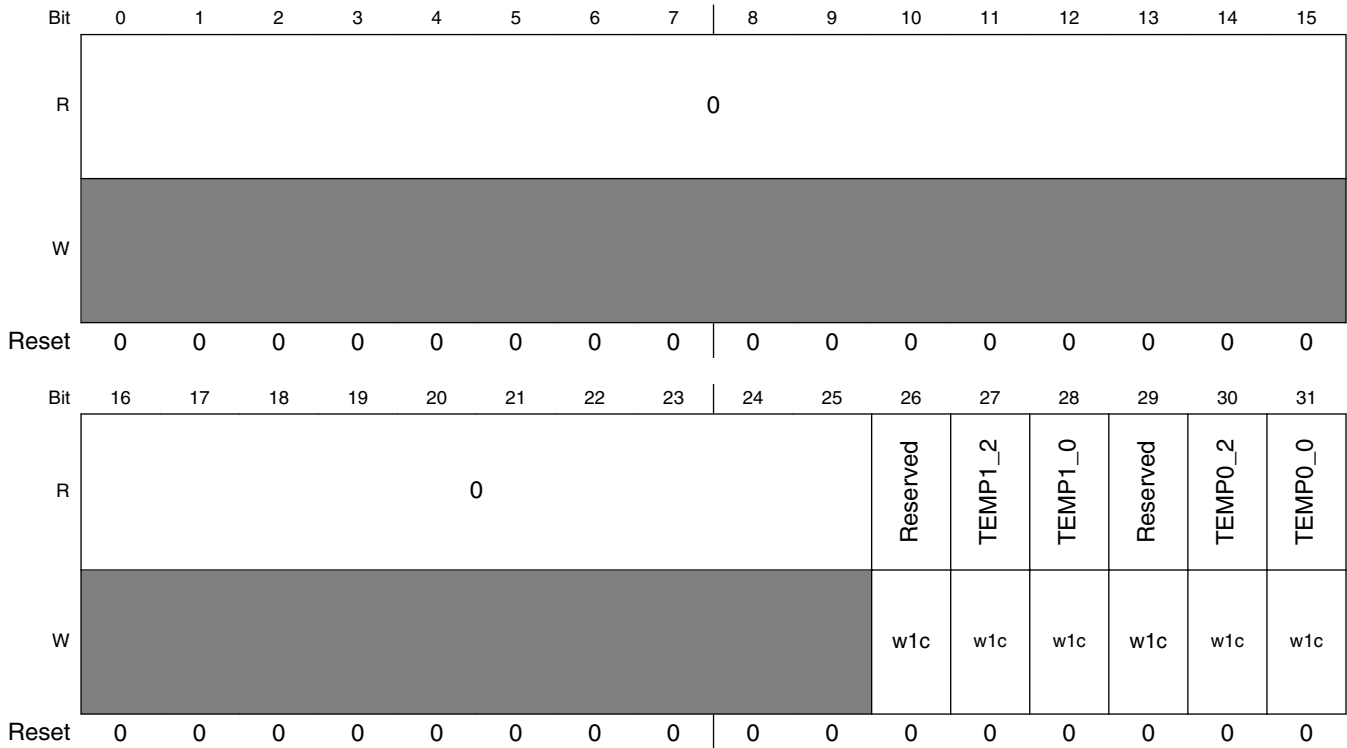
PMC_TS_IER field descriptions (continued)

Field	Description
	0 No interrupt. 1 Interrupt occurs.
28 TS1_0IE	Temperature Sensor 1 input 0 Interrupt Enable This bit determines whether an interrupt is seen by the system when the temperature falls below its corresponding threshold. 0 No interrupt. 1 Interrupt occurs.
29 Reserved	Reserved This field is reserved.
30 TS0_2IE	Temperature Sensor 0 input 2 Interrupt Enable This bit determines whether an interrupt is seen by the system when the temperature rises above its corresponding threshold (flag 2). 0 No interrupt. 1 Interrupt occurs.
31 TS0_0IE	Temperature Sensor 0 input 0 Interrupt Enable This bit determines whether an interrupt is seen by the system when the temperature falls below its corresponding threshold. 0 No interrupt. 1 Interrupt occurs.

54.2.2 Temperature Event Status Register (PMC_ESR_TD)

This register indicates the state of the two Temperature Sensor modules' signals. When a temperature event occurs, the corresponding flag bit sets to 1. To clear a flag bit to 0, write 1 to it.

Address: 0h base + 8Ch offset = 8Ch



PMC_ESR_TD field descriptions

Field	Description
0–25 Reserved	Reserved This field is reserved. This read-only field is reserved and always has the value 0.
26 Reserved	Reserved This field is reserved.
27 TEMP1_2	Temperature Sensor 1 event flag 2 This bit is associated with the temperature for the high temperature sensor 2 point (150°C). It sets to 1 when the temperature exceeds its corresponding threshold, and it clears to 0 when 1 is written to this bit location. When this bit is 1: If the TS_IER[TS1_2IE] bit is also 1, an interrupt is sent to the CPU.

Table continues on the next page...

PMC_ESR_TD field descriptions (continued)

Field	Description
	- If REE_TD[TEMP1_2] is also 1, a system reset is generated, which clears TS_IER[TS1_2IE] and negates the interrupt request. - If REE_TD[TEMP1_2] is 1 <i>and</i> RES_TD[TEMP1_2] is 0, a POR is generated. NOTE: If this flag is cleared to 0 while the over-temperature condition still exists, the flag will not automatically reassert to 1 until after the device falls below the corresponding over-temperature threshold. 0 Currently no occurrence. 1 Temperature occurrence detected.
28 TEMP1_0	Temperature Sensor 1 event flag 0 This bit is associated with the temperature for the cold temperature sensor 0 point (–40°C). It sets to 1 when the temperature falls below its threshold, and it clears to 0 when 1 is written to this bit location. When this bit is 1: - If the TS_IER[TS1_0IE] bit is also 1, an interrupt is sent to the CPU. - If REE_TD[TEMP1_0] is also 1, a system reset is generated, which clears TS_IER[TS1_0IE] and negates the interrupt request. - If REE_TD[TEMP1_0] is 1 <i>and</i> RES_TD[TEMP1_0] is 0, a POR is generated. NOTE: If this flag is cleared to 0 while the under-temperature condition still exists, the flag will not automatically reassert to 1 until after the device rises above the under-temperature threshold. 0 Currently no occurrence. 1 Temperature occurrence detected.
29 Reserved	Reserved This field is reserved.
30 TEMP0_2	Temperature Sensor 0 event flag 2 This bit is associated with the temperature for the high temperature sensor 2 point (150°C). It sets to 1 when the temperature exceeds its corresponding threshold, and it clears to 0 when 1 is written to this bit location. When this bit is 1: - If the TS_IER[TS0_2IE] bit is also 1, an interrupt is sent to the CPU. - If REE_TD[TEMP0_2] is also 1, a system reset is generated, which clears TS_IER[TS0_2IE] and negates the interrupt request. - If REE_TD[TEMP0_2] is 1 <i>and</i> RES_TD[TEMP0_2] is 0, a POR is generated. NOTE: If this flag is cleared to 0 while the over-temperature condition still exists, the flag will not automatically reassert to 1 until after the device falls below the corresponding over-temperature threshold. 0 Currently no occurrence. 1 Temperature occurrence detected.
31 TEMP0_0	Temperature Sensor 0 event flag 0 This bit is associated with the temperature for the cold temperature sensor 0 point (–40°C). It sets to 1 when the temperature falls below its threshold, and it clears to 0 when 1 is written to this bit location. When this bit is 1: If the TS_IER[TS0_0IE] bit is also 1, an interrupt is sent to the CPU.

Table continues on the next page...

PMC_ESR_TD field descriptions (continued)

Field	Description
	- If REE_TD[TEMPO_0] is also 1, a system reset is generated, which clears TS_IER[TS0_0IE] and negates the interrupt request. - If REE_TD[TEMPO_0] is 1 <i>and</i> RES_TD[TEMPO_0] is 0, a POR is generated. NOTE: If this flag is cleared to 0 while the under-temperature condition still exists, the flag will not automatically reassert to 1 until after the device rises above the under-temperature threshold. 0 Currently no occurrence. 1 Temperature occurrence detected.

54.2.3 Temperature Reset Event Enable register (PMC_REE_TD)

This register controls whether the temperature detect signal event causes a reset. If the desired flag bit is enabled, the temperature detect event can cause a reset to be generated when the selected temperature passes the trigger event.

NOTE

These bits can be loaded from a DCF record in the UTEST portion of the flash memory. When they are loaded from a DCF record, they cannot be changed by a register write.

Address: 0h base + 90h offset = 90h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	
R	0										Reserved		TEMP1_2	TEMP1_0	Reserved	TEMP0_2	TEMP0_0
W											Reserved		TEMP1_2	TEMP1_0	Reserved	TEMP0_2	TEMP0_0
Reset	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	

PMC_REE_TD field descriptions

Field	Description
0–25 Reserved	Reserved This field is reserved. This read-only field is reserved and always has the value 0.
26 Reserved	Reserved

Table continues on the next page...

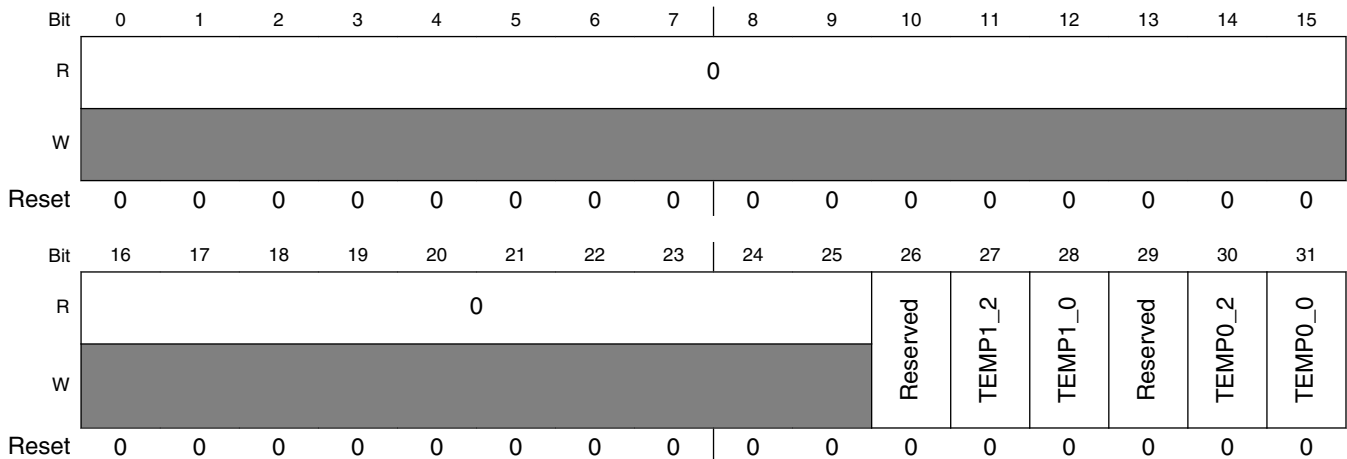
PMC_REE_TD field descriptions (continued)

Field	Description
	This field is reserved.
27 TEMP1_2	Temperature sensor 1 detect assertion reset enable 2 This bit defines whether a temperature detect assertion can generate a system reset. When this bit is 1, the RES_TD[TEMP1_2] bit determines whether the assertion causes a POR. 0 Disabled. Temperature Sensor detect does not cause system reset. 1 Enabled. Temperature Sensor detect can cause system reset.
28 TEMP1_0	Temperature sensor 1 detect assertion reset enable 0 This bit defines whether a temperature detect assertion can generate a system reset. When this bit is 1, the RES_TD[TEMP1_0] bit determines whether the assertion causes a POR. 0 Disabled. Temperature Sensor detect does not cause system reset. 1 Enabled. Temperature Sensor detect can cause system reset.
29 Reserved	Reserved This field is reserved.
30 TEMP0_2	Temperature sensor 0 detect assertion reset enable 2 This bit defines whether a temperature detect assertion can generate a system reset. When this bit is 1, the RES_TD[TEMP0_2] bit determines whether the assertion causes a POR. 0 Disabled. Temperature Sensor detect does not cause system reset. 1 Enabled. Temperature Sensor detect can cause system reset.
31 TEMP0_0	Temperature sensor 0 detect assertion reset enable 0 This bit defines whether a temperature detect assertion can generate a system reset. When this bit is 1, the RES_TD[TEMP0_0] bit determines whether the assertion causes a POR. 0 Disabled. Temperature Sensor detect does not cause system reset. 1 Enabled. Temperature Sensor detect can cause system reset.

54.2.4 Temperature Reset Event Selection Register (PMC_RES_TD)

This register controls whether the temperature sensor detect signal event causes a POR or no reset. If the desired flag bit is enabled, the temperature sensor event causes either a POR or no reset to be generated when the selected temperature passes the trigger event.

Address: 0h base + 94h offset = 94h



PMC_RES_TD field descriptions

Field	Description
0–25 Reserved	Reserved This field is reserved. This read-only field is reserved and always has the value 0.
26 Reserved	Reserved This field is reserved.
27 TEMP1_2	TEMP1_2 Reset Event Select 0 POR generated. Temperature Sensor assertion causes a POR. 1 No reset generated. Temperature Sensor assertion does not cause a reset.
28 TEMP1_0	TEMP1_0 Reset Event Select 0 POR generated. Temperature Sensor assertion causes a POR. 1 No reset generated. Temperature Sensor assertion does not cause a reset.
29 Reserved	Reserved This field is reserved.
30 TEMP0_2	TEMP0_2 Reset Event Select 0 POR generated. Temperature Sensor assertion causes a POR. 1 No reset generated. Temperature Sensor assertion does not cause a reset.

Table continues on the next page...

PMC_RES_TD field descriptions (continued)

Field	Description
31 TEMP0_0	TEMP0_0 Reset Event Select 0 POR generated. Temperature Sensor assertion causes a POR. 1 No reset generated. Temperature Sensor assertion does not cause a reset.

54.2.5 Temperature Detector Configuration Register (PMC_CTL_TD)

Address: 0h base + 98h offset = 98h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	Reserved					Reserved	TS1_EN	0	Reserved					Reserved	TS0_EN
W																
Reset	0	0	0	0	0	0	1	1	0	0	0	0	0	0	1	1

PMC_CTL_TD field descriptions

Field	Description
0–16 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
17–21 Reserved	This field is reserved.
22 Reserved	This field is reserved.
23 TS1_EN	Temperature Sensor 1 Enable NOTE: If this bit is 0, all reset and interrupt events from Temperature Sensor 1 are disabled. 0 Disable Temperature Sensor 1 module. 1 Enable Temperature Sensor 1 module.
24 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
25–29 Reserved	This field is reserved.

Table continues on the next page...

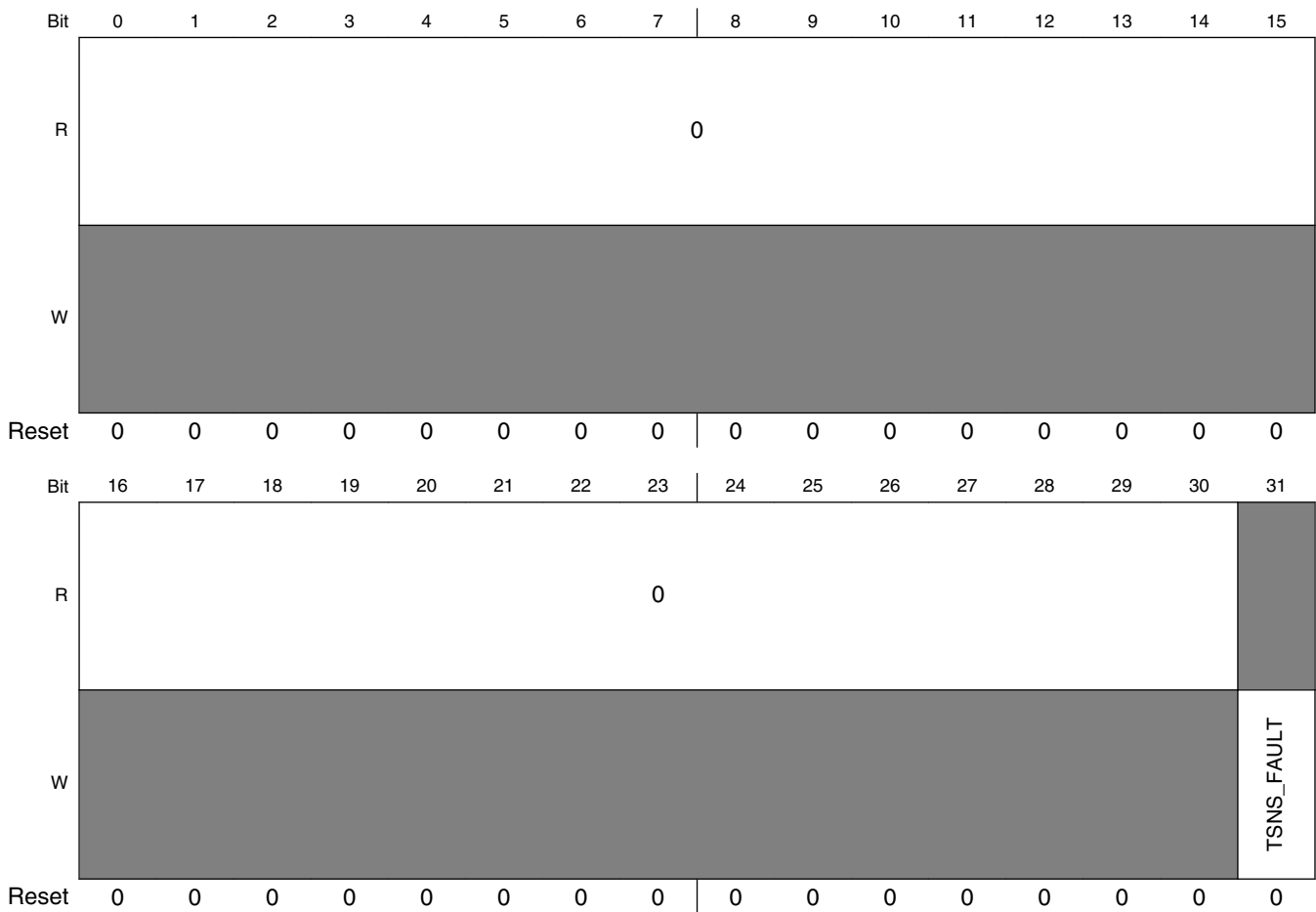
PMC_CTL_TD field descriptions (continued)

Field	Description
30 Reserved	This field is reserved.
31 TS0_EN	Temperature Sensor 0 Enable NOTE: If this bit is 0, all reset and interrupt events from Temperature Sensor 0 are disabled. 0 Disable Temperature Sensor 0 module. 1 Enable Temperature Sensor 0 module.

54.2.6 Temperature Sensor Fault Injection Register (PMC_TS_FIR)

This FCCU Fault Injection register enables user software to artificially generate FCCU faults corresponding to TEMP SENS. Writing 1 to one of these bits generates a single cycle pulsed fault to the FCCU block. The bit clears to 0 automatically in the next cycle.

Address: 0h base + A4h offset = A4h



PMC_TS_FIR field descriptions

Field	Description
0–30 Reserved	Reserved This field is reserved. This read-only field is reserved and always has the value 0.
31 TSNS_FAULT	Temperature Sensor Fault injection. A write to this bit sends a FCCU fault (single cycle) for TEMP SENSOR. 0 No fault is sent. 1 Single cycle fault injected.

54.2.7 Voltage Detect User Mode Test Register (PMC_SELF_TEST_UM_VD_REG)

This register is used for testing LVDs and HVDs during user mode.

NOTE

A PMC self test of any or all LVDs and HVDs can be interrupted by these reset sources and thereby become corrupted:

- Writing 1 to SIU_SRCR[SSR]
- JTAG reset
- FCCU reset

Due to this corruption, the PMC or FCCU might report incorrectly that a PMC self test of any or all LVDs and HVDs failed.

If software detects one of these reset sources when exiting reset and the PMC or FCCU reports a failed PMC self test of any or all LVDs and HVDs, execute the PMC self test again to confirm the failure.

Memory Map and Registers

Address: 0h base + B4h offset = B4h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0													ST_RESULT	ST_DONE	
W														w1c		
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0								ST_MODE		VD_ST_CTRL					
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

PMC_SELF_TEST_UM_VD_REG field descriptions

Field	Description
0–13 Reserved	Reserved This field is reserved. This read-only field is reserved and always has the value 0.
14 ST_RESULT	Self Test Result status Valid only when ST_DONE is 1. It is cleared on reset or by the host writing 1. 0 Self Test Failed 1 Self Test Passed
15 ST_DONE	Self Test Done status Self Test completed indication status. This bit is set to 1 (logic high) to indicate that a self test has completed. A value of 1 does not represent whether a self test has passed or failed. This bit is cleared to 0 automatically every time a self test is triggered. 0 Self test running 1 Self test completed

Table continues on the next page...

PMC_SELF_TEST_UM_VD_REG field descriptions (continued)

Field	Description																																
16–23 Reserved	Reserved This field is reserved. This read-only field is reserved and always has the value 0.																																
24–25 ST_MODE	Self Test Mode bits for testing LVDs and HVDs 00 Default 01 Software triggered self test. 10 Single VD test as indicated by the VD_ST_CTRL field. 11 Reserved																																
26–31 VD_ST_CTRL	Voltage Detect Self Test Control User mode Test bits for testing of LVDs and HVDs. Encoding values not listed are reserved. <table border="1"> <thead> <tr> <th>VD_ST_CTRL[5:0]</th><th>Test</th></tr> </thead> <tbody> <tr><td>000000</td><td>Functional Mode</td></tr> <tr><td>000001</td><td>Reserved</td></tr> <tr><td>000010</td><td>POR098_c</td></tr> <tr><td>000011</td><td>Reserved</td></tr> <tr><td>000100</td><td>Reserved</td></tr> <tr><td>000101</td><td>LVD_CORE_COLD</td></tr> <tr><td>000110</td><td>LVD_CORE_HOT</td></tr> <tr><td>000111</td><td>HVD_CORE</td></tr> <tr><td>001000</td><td>POR_HV</td></tr> <tr><td>001001</td><td>Reserved</td></tr> <tr><td>001010</td><td>LVD_HV</td></tr> <tr><td>001011</td><td>HVD_HV</td></tr> <tr><td>001100</td><td>LVD_FLASH</td></tr> <tr><td>001101</td><td>HVD_FLASH</td></tr> <tr><td>001110</td><td>LVD_IO</td></tr> </tbody> </table>	VD_ST_CTRL[5:0]	Test	000000	Functional Mode	000001	Reserved	000010	POR098_c	000011	Reserved	000100	Reserved	000101	LVD_CORE_COLD	000110	LVD_CORE_HOT	000111	HVD_CORE	001000	POR_HV	001001	Reserved	001010	LVD_HV	001011	HVD_HV	001100	LVD_FLASH	001101	HVD_FLASH	001110	LVD_IO
VD_ST_CTRL[5:0]	Test																																
000000	Functional Mode																																
000001	Reserved																																
000010	POR098_c																																
000011	Reserved																																
000100	Reserved																																
000101	LVD_CORE_COLD																																
000110	LVD_CORE_HOT																																
000111	HVD_CORE																																
001000	POR_HV																																
001001	Reserved																																
001010	LVD_HV																																
001011	HVD_HV																																
001100	LVD_FLASH																																
001101	HVD_FLASH																																
001110	LVD_IO																																

54.2.8 FCCU Fault Injection Register (PMC_FIR)

This FCCU Fault Injection register enables user software to artificially generate FCCU faults corresponding to various events, namely LVD, HVD, TEMP SENS and Self Test. A write of 1 to these bits generates a single-cycle pulsed fault to the FCCU block. This bit is automatically cleared to 0 in the next cycle.

Address: 0h base + B8h offset = B8h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0												RESERVED			
W													STEST_FAULT		HVD_FAULT	LVD_FAULT
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

PMC_FIR field descriptions

Field	Description
0–27 Reserved	Reserved This field is reserved. This read-only field is reserved and always has the value 0.
28 STEST_FAULT	A write of 1 to this bit sends a FCCU fault (single cycle) for LVD SELF TEST. This is a self clearing bit. 0 No fault is sent 1 Single cycle fault injected.
29 RESERVED	Reserved. This field is reserved. Reserved. Read to this bit will return always zero.
30 HVD_FAULT	High Voltage Detect Fault injection. A write of 1 to this bit sends a FCCU fault (single cycle) for HVD. This is a self clearing bit. 0 No fault is sent 1 Single cycle fault injected.
31 LVD_FAULT	Low Voltage Detect Fault injection. A write of 1 to this bit sends a FCCU fault (single cycle) for LVD. This is a self clearing bit. 0 No fault is sent 1 Single cycle fault injected.

54.2.9 LVD/HVD Interrupt Enable Control Register (PMC_LVD_HVD_INTR_ENB)

This register enables/disables interrupts of PMC VD events.

NOTE

Writes to this register must occur in two steps:

1. Set PMC_ENABLE in this register to 1.
2. In another write operation, program the other fields of the register.

Address: 0h base + BCh offset = BCh

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R									0							
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Memory Map and Registers

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0								Reserved		HVD_FLASH	Reserved	HVD_HV	HVD_CORE	Reserved	LVD_CORE_COLD
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

PMC_LVD_HVD_INTR_ENB field descriptions

Field	Description
0 PMC_ENABLE	PMC Enable Control This bit determines whether any of the PMC bits can be written or not. 0 No control bit can be written in this(PMC enable control) register.To do a write on this register, first enable this bit and then start witting on other control bits. 1 This Control register can be written.
1–23 Reserved	Reserved This field is reserved. This read-only field is reserved and always has the value 0.
24–25 Reserved	Reserved This field is reserved. When writing to this field, always write the reset value.
26 HVD_FLASH	HVD Flash Enable/Disable This bit enable/disable the generation of Interrupt due to an event on this supply. 0 Interrupt is disabled. 1 Interrupt is Enabled.
27 Reserved	Reserved This field is reserved. When writing to this field, always write the reset value.
28 HVD_HV	HVD_HV Enable/Disable This bit enable/disable the generation of Interrupt due to an event on this supply. 0 Interrupt is disabled. 1 Interrupt is Enabled.
29 HVD_CORE	HVD_CORE Enable/Disable This bit enable/disable the generation of Interrupt due to an event on this supply. 0 Interrupt is disabled. 1 Interrupt is Enabled.
30 Reserved	Reserved This field is reserved. When writing to this field, always write the reset value.

Table continues on the next page...

PMC_LVD_HVD_INTR_ENB field descriptions (continued)

Field	Description
31 LVD_CORE_ COLD	LVD Core Cold Enable/Disable This bit enable/disable the generation of Interrupt due to an event on this supply. 0 Interrupt is disabled. 1 Interrupt is Enabled.

54.2.10 LVD/HVD Event Status Register
(PMC_LVD_HVD_EVENT_STATUS)

This status register contains the latched status of the voltage detect events. These bits indicate the state of the voltage detect events, including whether a voltage level event has occurred at any time in the past but has not been cleared (via writing 1). These bits are read-only, and they are cleared by writing 1 to the corresponding ESR register bit.

Address: 0h base + C0h offset = C0h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0								Reserved	Reserved	HVD_FLASH	Reserved	HVD_HV	HVD_CORE	Reserved	LVD_CORE_COLD
W									w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

PMC_LVD_HVD_EVENT_STATUS field descriptions

Field	Description
0–23 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

Table continues on the next page...

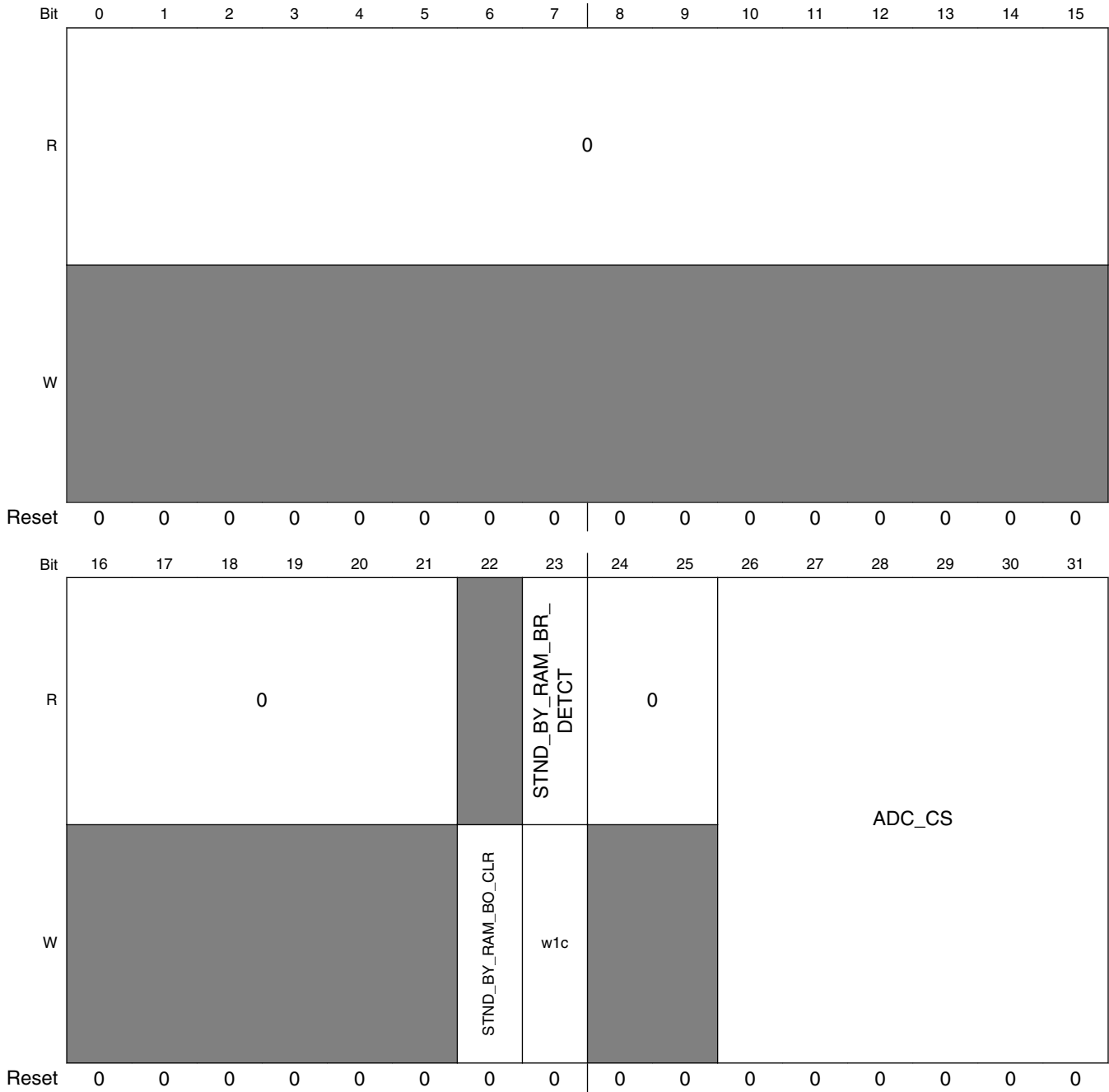
PMC_LVD_HVD_EVENT_STATUS field descriptions (continued)

Field	Description
24 Reserved	Reserved This field is reserved. When writing to this field, always write the reset value.
25 Reserved	Reserved This field is reserved. When writing to this field, always write the reset value.
26 HVD_FLASH	This bit controls the enable/disable on the LVD PLL supply. 0 No occurrence has happened. 1 An occurrence has happened.
27 Reserved	Reserved This field is reserved. When writing to this field, always write the reset value.
28 HVD_HV	HVD_HV Status Flag 0 No occurrence has happened. 1 An occurrence has happened.
29 HVD_CORE	HVD Core Status Flag 0 No occurrence has happened. 1 An occurrence has happened.
30 Reserved	Reserved This field is reserved. When writing to this field, always write the reset value.
31 LVD_CORE_COLD	LVD Core Cold Status Flag 0 No occurrence has happened. 1 An occurrence has happened.

54.2.11 ADC Channel Select Register (PMC_ADC_CS)

This register contains the SRAM Brownout control and the analog channel mux of the analog PMC module.

Address: 0h base + C4h offset = C4h



PMC_ADC_CS field descriptions

Field	Description																																														
0–21 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.																																														
22 STND_BY_ RAM_BO_CLR	Standby RAM Brownout Clear A write to this bit generates a 1-clock-wide pulse. It is a self-clearing pulse.																																														
23 STND_BY_ RAM_BR_ DETCT	Standby RAM Brownout Detect This bit is the detection status flag for the Standby RAM Brownout Detection circuit. When Standby RAM logic detects a brownout event, this bit is set to 1. To clear this flag, two separate write accesses are required: first write 1 to STND_BY_RAM_BO_CLR, and then write 1 to STND_BY_RAM_BR_DETCT. 0 Currently no occurrence. 1 Standby RAM Brownout occurrence detected.																																														
24–25 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.																																														
26–31 ADC_CS	Analog-to-Digital Converter Channel Select. These bits are used to control the analog PMC's test mux going to the ADC. This is to probe the internal node voltages of the PMC for testing, calibration and characterization purposes by one of the ADCs of the SoC. Encoding values not listed are reserved. <table border="1"> <thead> <tr> <th>ADC_CS[5:0]</th><th>Function</th></tr> </thead> <tbody> <tr><td>000000</td><td>ADC channel off</td></tr> <tr><td>011110</td><td>Reserved</td></tr> <tr><td>011101</td><td>Reserved</td></tr> <tr><td>011100</td><td>POR098_c sense</td></tr> <tr><td>011011</td><td>POR098_c reference</td></tr> <tr><td>011010</td><td>Reserved</td></tr> <tr><td>011001</td><td>Reserved</td></tr> <tr><td>011000</td><td>Reserved</td></tr> <tr><td>010111</td><td>Reserved</td></tr> <tr><td>010110</td><td>LVD_CORE_COLD sense</td></tr> <tr><td>010101</td><td>LVD_CORE_COLD reference</td></tr> <tr><td>010100</td><td>LVD_CORE_HOT sense</td></tr> <tr><td>010011</td><td>LVD_CORE_HOT reference</td></tr> <tr><td>010010</td><td>HVD_CORE sense</td></tr> <tr><td>010001</td><td>HVD_CORE reference</td></tr> <tr><td>010000</td><td>POR_HV divider tap</td></tr> <tr><td>001111</td><td>POR_HV reference (PMC band gap 1200 mV)</td></tr> <tr><td>001110</td><td>Reserved</td></tr> <tr><td>001101</td><td>Reserved</td></tr> <tr><td>001100</td><td>LVD_HV divider tap</td></tr> <tr><td>001011</td><td>LVD_HV reference (PMC band gap 1200 mV)</td></tr> <tr><td>001010</td><td>HVD_HV divider tap</td></tr> </tbody> </table>	ADC_CS[5:0]	Function	000000	ADC channel off	011110	Reserved	011101	Reserved	011100	POR098_c sense	011011	POR098_c reference	011010	Reserved	011001	Reserved	011000	Reserved	010111	Reserved	010110	LVD_CORE_COLD sense	010101	LVD_CORE_COLD reference	010100	LVD_CORE_HOT sense	010011	LVD_CORE_HOT reference	010010	HVD_CORE sense	010001	HVD_CORE reference	010000	POR_HV divider tap	001111	POR_HV reference (PMC band gap 1200 mV)	001110	Reserved	001101	Reserved	001100	LVD_HV divider tap	001011	LVD_HV reference (PMC band gap 1200 mV)	001010	HVD_HV divider tap
ADC_CS[5:0]	Function																																														
000000	ADC channel off																																														
011110	Reserved																																														
011101	Reserved																																														
011100	POR098_c sense																																														
011011	POR098_c reference																																														
011010	Reserved																																														
011001	Reserved																																														
011000	Reserved																																														
010111	Reserved																																														
010110	LVD_CORE_COLD sense																																														
010101	LVD_CORE_COLD reference																																														
010100	LVD_CORE_HOT sense																																														
010011	LVD_CORE_HOT reference																																														
010010	HVD_CORE sense																																														
010001	HVD_CORE reference																																														
010000	POR_HV divider tap																																														
001111	POR_HV reference (PMC band gap 1200 mV)																																														
001110	Reserved																																														
001101	Reserved																																														
001100	LVD_HV divider tap																																														
001011	LVD_HV reference (PMC band gap 1200 mV)																																														
001010	HVD_HV divider tap																																														

Table continues on the next page...

PMC_ADC_CS field descriptions (continued)

Field	Description	
	ADC_CS[5:0]	Function
	001001	HVD_HV reference
	001000	LVD_FLASH divider tap
	000111	LVD_FLASH reference (PMC band gap 1200 mV)
	000110	HVD_FLASH divider tap
	000101	HVD_FLASH reference
	000100	LVD_IO divider tap
	000011	LVD_IO reference

54.2.12 Reset Event Enable Register (PMC_REE)

This Reset Event Enable Register controls whether or not the voltage detect signal event causes a reset. If the desired flag bit is enabled, the voltage detect event causes a POR to be generated when the selected voltage passes the trigger event. These bits are loaded from the flash memory during the boot sequence. If the bits loaded from flash memory are programmed to be enabled (set), these bits cannot be disabled (cleared).

NOTE

This register is cleared on POR or a reset due to RESET pin assertion or FOSU failure indication.

CAUTION

Except when they occur during a PMC self test, the following VD events *always* result in a POR. The result of these VD events cannot be programmed:

- POR098_c
- LVD_CORE_HOT
- POR_HV
- LVD_HV
- LVD_FLASH
- LVD_IO

Address: 0h base + D0h offset = D0h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0								Reserved		HVD_FLASH	Reserved	HVD_HV	HVD_CORE	Reserved	LVD_CORE_COLD
W																
Reset	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1

PMC_REE field descriptions

Field	Description
0–23 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
24–25 Reserved	Reserved This field is reserved. When writing to this field, always write the reset value.
26 HVD_FLASH	Reset Enable/Disable Control This particular bit enable or disable the generation of reset due to an event on this supply. 0 Reset is disabled, no reset will be generated even if this event occurred. 1 Reset is enabled. A reset will be generated if an event is detected on this supply
27 Reserved	Reserved This field is reserved. When writing to this field, always write the reset value.
28 HVD_HV	Reset Enable/Disable Control This particular bit enable or disable the generation of reset due to an event on this supply. 0 Reset is disabled, no reset will be generated even if this event occurred. 1 Reset is enabled. A reset will be generated if an event is detected on this supply
29 HVD_CORE	Reset Enable/Disable Control This particular bit enable or disable the generation of reset due to an event on this supply. 0 Reset is disabled, no reset will be generated even if this event occurred. 1 Reset is enabled. A reset will be generated if an event is detected on this supply
30 Reserved	Reserved This field is reserved.

Table continues on the next page...

PMC_REE field descriptions (continued)

Field	Description
	When writing to this field, always write the reset value.
31 LVD_CORE_ COLD	Reset Enable/Disable Control This particular bit enable or disable the generation of reset due to an event on this supply. 0 Reset is disabled, no reset will be generated even if there is event occurred on this supply. 1 Reset is enabled. A reset will be generated if an event is detected on this supply.

54.3 Functional description

54.3.1 Analog PMC interface

The digital block uses the analog PMC interface to configure factory-defined trim values for the regulators and LVDs that are in the analog block.

54.3.2 Temperature Sensor interface logic

The digital PMC block provides control and configuration information, including module enables, to the two Temperature Sensors.

54.3.3 POR phase gates

The interface with the SIU module is used during the power-on sequence and startup. It provides various phase gates, thereby enabling the SIU to properly complete the reset sequence.

It also provides reset sources based on LVD and HVD events and their enable/disable control through the REE register.

The temperature sensor provides another reset source, based on the under/over temperature events and their control through the REE_TD and RES_TD registers.

54.3.4 Flash memory interface to DCF

During power up, trim values for the regulators, LVDs, and HVDs are read from flash memory by the SSCM and then provided to the digital PMC via the DCF interface.

The reset event enable (REE) configurations that are stored in flash memory are also provided to the PMC via the DCF interface. These REE configurations are used, by default, to configure which programmable voltage detection (VD) events generate a POR.

54.3.5 FCCU interface

The digital block generates various fault indications to the FCCU block:

- Separate, corresponding, active-high fault indications are generated whenever an LVD or HVD defect is detected in the analog block.
- A fault indication is generated when a self test task results in a failure.
- A fault condition is generated whenever over-temperature or under-temperature events occur in the temperature sensor.

The figure below show the exact logic for the generation fault indications to FCCU block.

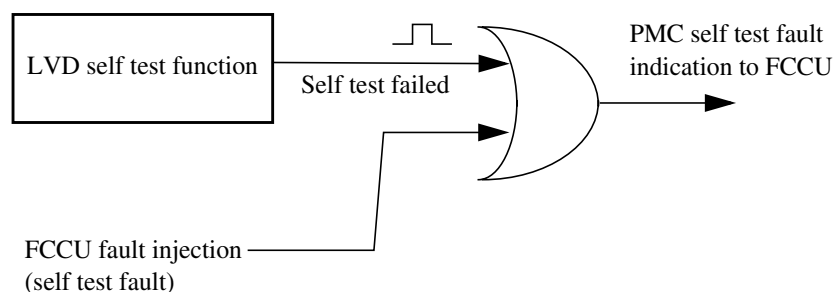


Figure 54-13. Fault generation to FCCU

See [FCCU Fault Injection Register \(PMC_FIR\)](#) for details.

54.3.6 LVD/HVD self test mechanism

The PMC implements an LVD/HVD self test mechanism for use in satisfying safety goals. The digital block assists and controls the analog block during execution of the self test.

Currently, three self test modes are supported by the analog block. Select the mode using the SELF_TEST_UM_VD_REG[ST_MODE] field.

The self test status and result are provided in the SELF_TEST_UM_VD_REG register's ST_DONE and ST_RESULT fields, respectively.

NOTE

Self tests must execute with the FM_PER_CLK frequency at 8 MHz.

54.3.6.1 Default mode

The self test always occurs during startup (power on). The self test starts after the SSCM has completed reads from the flash memory. A counter is provided that starts from zero and increments by one after a factory-defined time gap. The counter value is the source for the self test output signal, which is provided to the analog block.

Meanwhile, the LVD/HVD indication from the analog block is latched in the SELF_TEST_UM_VD_REG register. At the end of the self test, after all of the LVDs/HVDs have been tested, the SELF_TEST_UM_VD_REG[ST_DONE] status bit is set to 1 and all bits in the SELF_TEST_UM_VD_REG register are read to see if the value of each is 1.

- If one or more bits is not set to 1, a fault indication to the FCCU block is generated, and the SELF_TEST_UM_VD_REG[ST_RESULT] field is set to 0.
- If each of the register bits is set to 1, the test is considered passed, the SELF_TEST_UM_VD_REG[ST_RESULT] bit is set to 1, and no fault indications are provided to the FCCU.

NOTE

The total run time of the self test is 300 μ s (factory-defined default).

54.3.6.2 Software-triggered self test

This mode is entered by software by writing 01b to the SELF_TEST_UM_VD_REG[ST_MODE] field. The complete self test, as is performed in default mode, is executed. The difference is that the self test in this mode is initiated by the host while the self test in default mode is started automatically during power on.

54.3.6.3 Single voltage detect test

This mode is entered by software writing 10b to the SELF_TEST_UM_VD_REG[ST_MODE] field. The test completes after the prescribed time gap has elapsed (factory-defined default is 20 μ s). The PASS/FAIL result is then updated based on whether or not the corresponding VD output trips.

NOTE

During execution of the VD self test, all of the LVD and HVD outputs—including PORs, which are also under self test—are masked. Therefore, they have no effect on POR generation or any reset and/or fault generation.

Chapter 55

JTAG Controller (JTAGC)

55.1 Chip-specific JTAGC information

This chip has one instance of the JTAGC module.

55.1.1 JTAG ID

The JTAG ID depends on the version of the chip.

Table 55-1. JTAG ID by chip version

Maskset	JTAG ID
1N45H	0837_701Dh
2N45H	0837_701Dh
3N45H	1837_701Dh

For details, see [Device identification register](#).

55.2 Introduction

The JTAGC block provides the means to test chip functionality and connectivity while remaining transparent to system logic when not in test mode. Testing is performed via a boundary scan technique, as defined in the IEEE 1149.1-2001 standard. All data input to and output from the JTAGC block is communicated in serial format.

55.2.1 Block diagram

The following is a simplified block diagram of the JTAG Controller (JTAGC) block. Refer to the chip-specific configuration information as well as [Register description](#) for more information about the JTAGC registers.

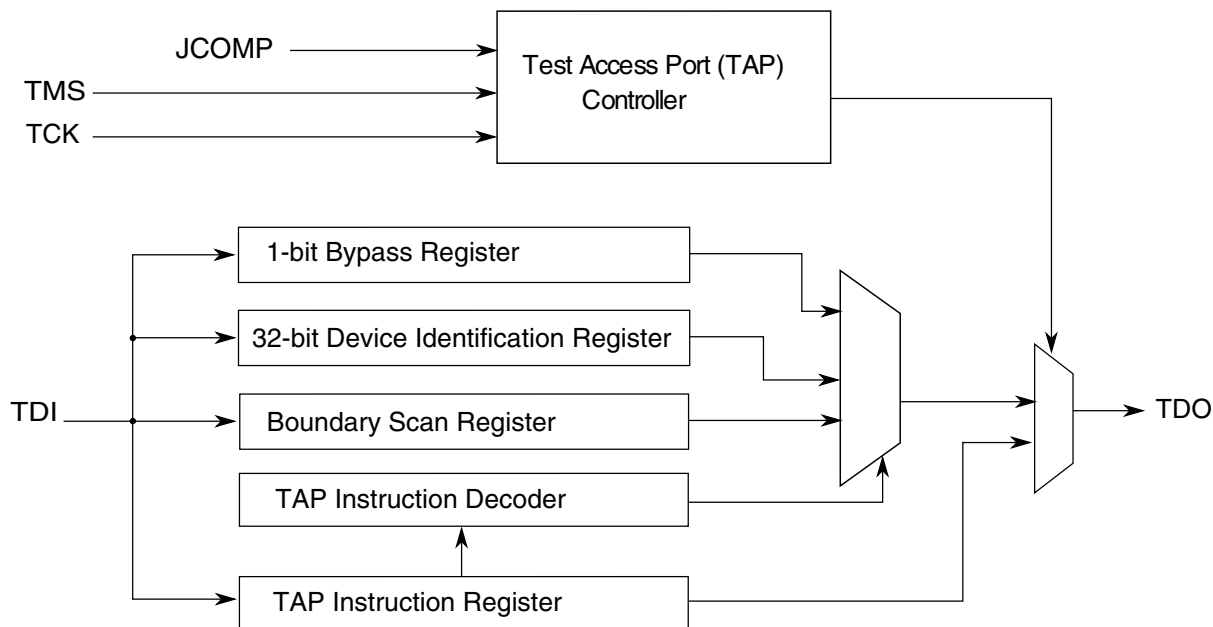


Figure 55-1. JTAG (IEEE 1149.1) block diagram

55.2.2 Features

The JTAGC block is compliant with the IEEE 1149.1-2001 standard, and supports the following features:

- IEEE 1149.1-2001 Test Access Port (TAP) interface
 - 4 pins (TDI, TMS, TCK, and TDO)
- JCOMP input that provides reset control
- Instruction register that supports several IEEE 1149.1-2001 defined instructions as well as several public and private device-specific instructions. Refer to [Table 55-5](#) for a list of supported instructions.
- Sharing of the TAP with other TAP controllers via ACCESS_AUX_x instructions
- JTAG_PASSWORD register

- Bypass register, boundary scan register, and device identification register.
- TAP controller state machine that controls the operation of the data registers, instruction register and associated circuitry.

55.2.3 Modes of operation

The JTAGC block uses JCOMP and a power-on reset indication as its primary reset signals. Several IEEE 1149.1-2001 defined test modes are supported, as well as a bypass mode.

55.2.3.1 Reset

The JTAGC block is placed in reset when either power-on reset is asserted, JCOMP is negated, or the TMS input is held high for enough consecutive rising edges of TCK to sequence the TAP controller state machine into the Test-Logic-Reset state. Holding TMS high for five consecutive rising edges of TCK guarantees entry into the Test-Logic-Reset state regardless of the current TAP controller state. Asserting power-on reset or setting JCOMP to a value other than the value required to enable the JTAGC block results in asynchronous entry into the reset state. While in reset, the following actions occur:

- The TAP controller is forced into the Test-Logic-Reset state, thereby disabling the test logic and allowing normal operation of the on-chip system logic to continue unhindered
- The instruction register is loaded with the IDCODE instruction

55.2.3.2 IEEE 1149.1-2001 defined test modes

The JTAGC block supports several IEEE 1149.1-2001 defined test modes. A test mode is selected by loading the appropriate instruction into the instruction register while the JTAGC is enabled. Supported test instructions include EXTEST, HIGHZ, CLAMP, SAMPLE and SAMPLE/PRELOAD. Each instruction defines the set of data register(s) that may operate and interact with the on-chip system logic while the instruction is current. Only one test data register path is enabled to shift data between TDI and TDO for each instruction.

The boundary scan register is enabled for serial access between TDI and TDO when the EXTEST, SAMPLE or SAMPLE/PRELOAD instructions are active. The single-bit bypass register shift stage is enabled for serial access between TDI and TDO when the BYPASS, HIGHZ, CLAMP or reserved instructions are active. The functionality of each test mode is explained in more detail in [JTAGC block instructions](#).

55.2.3.3 Bypass mode

When no test operation is required, the BYPASS instruction can be loaded to place the JTAGC block into bypass mode. While in bypass mode, the single-bit bypass shift register is used to provide a minimum-length serial path to shift data between TDI and TDO.

55.3 External signal description

The JTAGC consists of a set of signals that connect to off chip development tools and allow access to test support functions. The JTAGC signals are outlined in the following table and described in the following sections.

Table 55-2. JTAG signal properties

Name	I/O	Function	Reset State
TCK	Input	Test Clock	Weak pulldown
TDI	Input	Test Data In	Weak pullup
TDO	Output	Test Data Out	High Z ¹
TMS	Input	Test Mode Select	Weak pullup
JCOMP	Input	JTAG Compliancy	Weak pulldown

1. TDO output buffer enable is negated when the JTAGC is not in the Shift-IR or Shift-DR states. A weak pull may be implemented at the TDO pad for use when JTAGC is inactive.

55.3.1 TCK—Test clock input

Test Clock Input (TCK) is an input pin used to synchronize the test logic and control register access through the TAP.

55.3.2 TDI—Test data input

Test Data Input (TDI) is an input pin that receives serial test instructions and data. TDI is sampled on the rising edge of TCK.

55.3.3 TDO—Test data output

Test Data Output (TDO) is an output pin that transmits serial output for test instructions and data. TDO is three-stateable and is actively driven only in the Shift-IR and Shift-DR states of the TAP controller state machine, which is described in [TAP controller state machine](#).

55.3.4 TMS—Test mode select

Test Mode Select (TMS) is an input pin used to sequence the IEEE 1149.1-2001 test control state machine. TMS is sampled on the rising edge of TCK.

55.3.5 JCOMP—JTAG compliancy

The JCOMP signal provides IEEE 1149.1-2001 compatibility and provides the ability to share the TAP. The JTAGC TAP controller is enabled when JCOMP is set to the JTAGC enable encoding, otherwise the JTAGC TAP controller remains in reset.

55.4 Register description

This section provides a detailed description of the JTAGC block registers accessible through the TAP interface, including data registers and the instruction register. Individual bit-level descriptions and reset states of each register are included. These registers are not memory-mapped and can only be accessed through the TAP.

55.4.1 Instruction register

The JTAGC block uses a 5-bit instruction register as shown in the following figure. The instruction register allows instructions to be loaded into the block to select the test to be performed or the test data register to be accessed or both. Instructions are shifted in through TDI while the TAP controller is in the Shift-IR state, and latched on the falling edge of TCK in the Update-IR state. The latched instruction value can only be changed in

the Update-IR and Test-Logic-Reset TAP controller states. Synchronous entry into the Test-Logic-Reset state results in the IDCODE instruction being loaded on the falling edge of TCK. Asynchronous entry into the Test-Logic-Reset state results in asynchronous loading of the IDCODE instruction. During the Capture-IR TAP controller state, the instruction shift register is loaded with the value 00001b , making this value the register's read value when the TAP controller is sequenced into the Shift-IR state.

	4	3	2	1	0
R	0	0	0	0	1
W	Instruction Code				
Reset:	0	0	0	0	1

Figure 55-2. Instruction register

55.4.2 Bypass register

The bypass register is a single-bit shift register path selected for serial data transfer between TDI and TDO when the BYPASS, CLAMP, HIGHZ or reserve instructions are active. After entry into the Capture-DR state, the single-bit shift register is set to a logic 0. Therefore, the first bit shifted out after selecting the bypass register is always a logic 0.

55.4.3 Device identification register

The device identification (JTAG ID) register, shown in the following figure, allows the revision number, part number, manufacturer, and design center responsible for the design of the part to be determined through the TAP. The device identification register is selected for serial data transfer between TDI and TDO when the IDCODE instruction is active. Entry into the Capture-DR state while the device identification register is selected loads the IDCODE into the shift register to be shifted out on TDO in the Shift-DR state. No action occurs in the Update-DR state.

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	Part Revision Number				Design Center						Part Identification Number					
W																
Reset	PRN				DC						PIN					
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	Part Identification Number				Manufacturer Identity Code										1	
W																
Reset	PIN (contd.)				MIC										1	

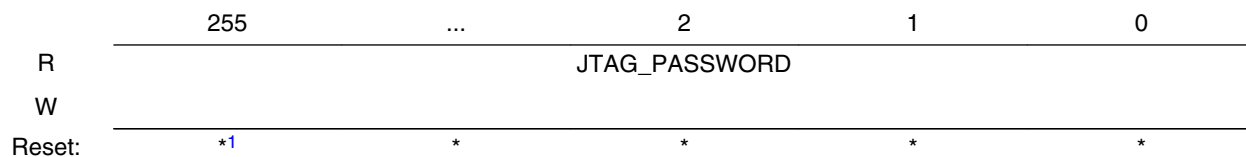
The following table describes the device identification register functions.

Table 55-3. Device identification register field descriptions

Field	Description
PRN	Part Revision Number. Contains the revision number of the part. Value is 0x0 for masksets 1N45H and 2N45H, 0x1 for maskset 3N45H.
DC	Design Center. Indicates the design center. Value is 0x20.
PIN	Part Identification Number. Contains the part number of the device. 0x377.
MIC	Manufacturer Identity Code. Contains the reduced Joint Electron Device Engineering Council (JEDEC) ID. Value is 0x00E.
IDCODE ID	IDCODE Register ID. Identifies this register as the device identification register and not the bypass register. Always set to 1.

55.4.4 JTAG_PASSWORD register

The JTAG_PASSWORD register is a 256-bit shift register path from TDI to TDO selected when the ENABLE_JTAG_PASSWORD instruction is active. The default reset value of the JTAG_PASSWORD register is 256'b0. The JTAG_PASSWORD register transfers its value to a parallel hold register on the rising edge of TCK when the TAP controller state machine is in the Update-DR state. Once the ENABLE_JTAG_PASSWORD instruction is executed, the register value remains valid until a JTAG reset occurs. The operation of this register is described in the security documentation.



1. The reset value of JTAG_PASSWORD is 256 'b0.

The following table describes the JTAG_PASSWORD register functions.

Table 55-4. JTAG_PASSWORD register field descriptions

Field	Description
JTAG_PASSWORD	JTAG Password. The JTAG_PASSWORD bits are used to provide the JTAG password for security.

55.4.5 Boundary scan register

The boundary scan register is connected between TDI and TDO when the EXTEST, SAMPLE or SAMPLE/PRELOAD instructions are active. It is used to capture input pin data, force fixed values on output pins, and select a logic value and direction for bidirectional pins. Each bit of the boundary scan register represents a separate boundary scan register cell, as described in the IEEE 1149.1-2001 standard and discussed in [Boundary scan](#). The size of the boundary scan register and bit ordering is device-dependent and can be found in the device BSDL file.

55.5 Functional description

This section explains the JTAGC functional description.

55.5.1 JTAGC reset configuration

While in reset, the TAP controller is forced into the Test-Logic-Reset state, thus disabling the test logic and allowing normal operation of the on-chip system logic. In addition, the instruction register is loaded with the IDCODE instruction.

55.5.2 IEEE 1149.1-2001 (JTAG) Test Access Port

The JTAGC block uses the IEEE 1149.1-2001 TAP for accessing registers. This port can be shared with other TAP controllers on the MCU. Ownership of the port is determined by the value of the currently loaded instruction. For more detail on TAP sharing via JTAGC instructions refer to [ACCESS_AUX_x instructions](#).

Data is shifted between TDI and TDO through the selected register starting with the least significant bit, as illustrated in the following figure. This applies for the instruction register, test data registers, and the bypass register.

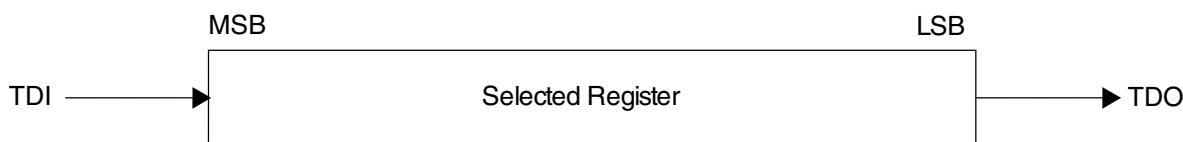
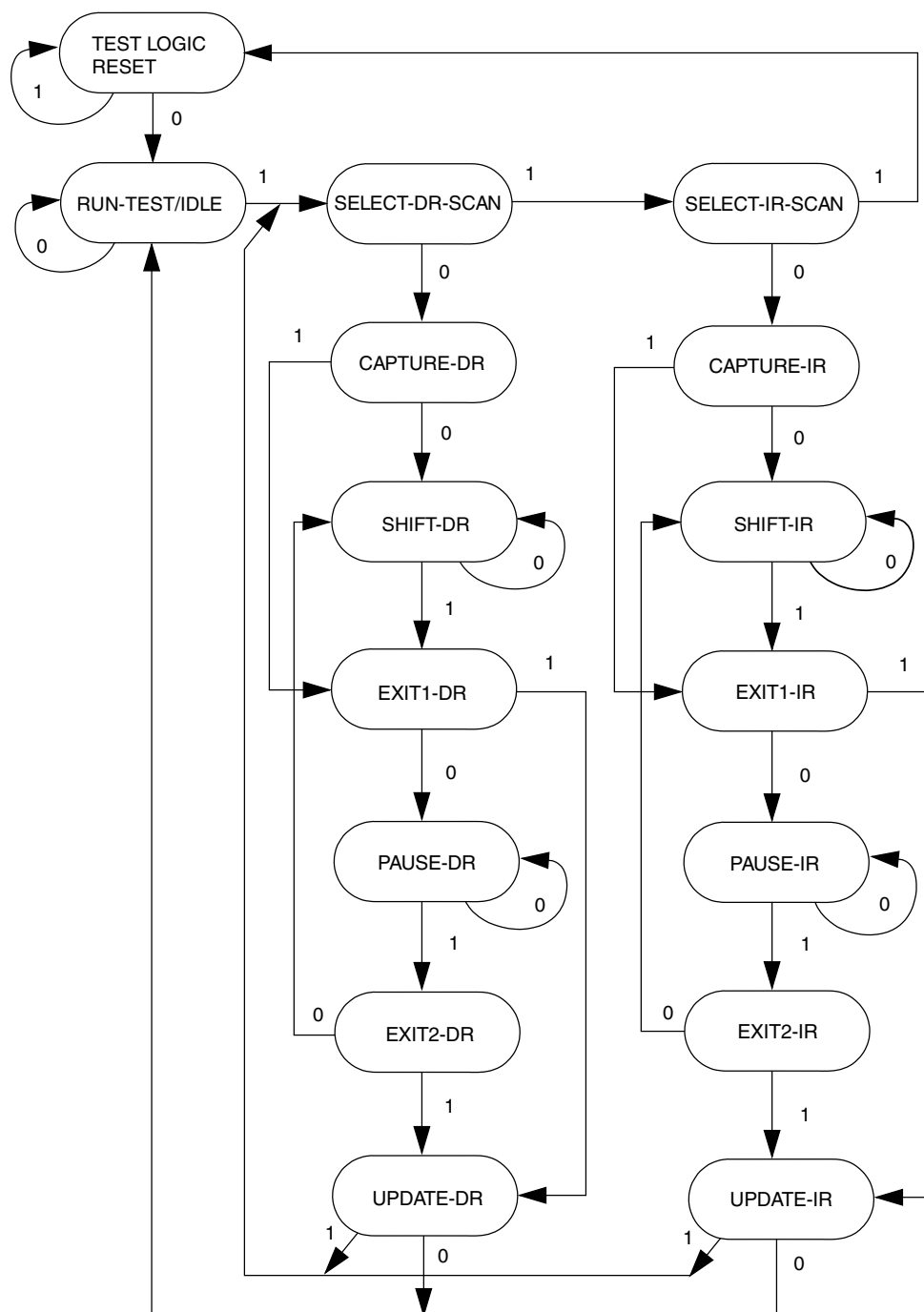


Figure 55-3. Shifting data through a register

55.5.3 TAP controller state machine

The TAP controller is a synchronous state machine that interprets the sequence of logical values on the TMS pin. The following figure shows the machine's states. The value shown next to each state is the value of the TMS signal sampled on the rising edge of the TCK signal. As the following figure shows, holding TMS at logic 1 while clocking TCK through a sufficient number of rising edges also causes the state machine to enter the Test-Logic-Reset state.



The value shown adjacent to each state transition in this figure represents the value of TMS at the time of a rising edge of TCK.

Figure 55-4. IEEE 1149.1-2001 TAP controller finite state machine

55.5.3.1 Enabling the TAP controller

The JTAGC TAP controller is enabled by setting JCOMP to a logic 1 value.

55.5.3.2 Selecting an IEEE 1149.1-2001 register

Access to the JTAGC data registers is achieved by loading the instruction register with any of the JTAGC block instructions while the JTAGC is enabled. Instructions are shifted in via the Select-IR-Scan path and loaded in the Update-IR state. At this point, all data register access is performed via the Select-DR-Scan path.

The Select-DR-Scan path is used to read or write the register data by shifting in the data (LSB first) during the Shift-DR state. When reading a register, the register value is loaded into the IEEE 1149.1-2001 shifter during the Capture-DR state. When writing a register, the value is loaded from the IEEE 1149.1-2001 shifter to the register during the Update-DR state. When reading a register, there is no requirement to shift out the entire register contents. Shifting may be terminated once the required number of bits have been acquired.

55.5.4 JTAGC block instructions

The JTAGC block implements the IEEE 1149.1-2001 defined instructions listed in the following table. This section gives an overview of each instruction; refer to the IEEE 1149.1-2001 standard for more details. All undefined opcodes are reserved.

Table 55-5. General 5-bit JTAG instructions

Instruction	Code[4:0]	Instruction Summary
IDCODE	00001	Selects device identification register for shift
SAMPLE/PRELOAD	00010	Selects boundary scan register for shifting, sampling, and preloading without disturbing functional operation
SAMPLE	00011	Selects boundary scan register for shifting and sampling without disturbing functional operation
EXTEST	00100	Selects boundary scan register and applies preloaded values to output pins. NOTE: Execution of this instruction asserts functional reset.
Factory debug reserved	00101	Intended for factory debug only
Factory debug reserved	00110	Intended for factory debug only
ENABLE_JTAG_PASSWORD	00111	Selects JTAG_PASSWORD register
HIGHZ	01001	Selects bypass register and three-states all output pins. NOTE: Execution of this instruction asserts functional reset.
Factory debug reserved	01010	Intended for factory debug only
CLAMP	01100	Selects bypass register and applies preloaded values to output pins. NOTE: Execution of this instruction asserts functional reset.

Table continues on the next page...

Table 55-5. General 5-bit JTAG instructions (continued)

Instruction	Code[4:0]	Instruction Summary
Factory debug reserved	01110	Intended for factory debug only
Factory debug reserved	01111	Intended for factory debug only
BYPASS	11111	Selects bypass register for data operations
Reserved ¹	All other opcodes	Decoded to select bypass register

1. The manufacturer reserves the right to change the decoding of reserved instruction codes in the future

The ACCESS_AUX instructions are described in the chip configuration debug information.

55.5.4.1 IDCODE instruction

IDCODE selects the 32-bit device identification register as the shift path between TDI and TDO. This instruction allows interrogation of the MCU to determine its version number and other part identification data. IDCODE is the instruction placed into the instruction register when the JTAGC block is reset.

55.5.4.2 SAMPLE/PRELOAD instruction

The SAMPLE/PRELOAD instruction has two functions:

- The SAMPLE portion of the instruction obtains a sample of the system data and control signals present at the MCU input pins and just before the boundary scan register cells at the output pins. This sampling occurs on the rising edge of TCK in the Capture-DR state when the SAMPLE/PRELOAD instruction is active. The sampled data is viewed by shifting it through the boundary scan register to the TDO output during the Shift-DR state. Both the data capture and the shift operation are transparent to system operation.
- The PRELOAD portion of the instruction initializes the boundary scan register cells before selecting the EXTEST or CLAMP instructions to perform boundary scan tests. This is achieved by shifting in initialization data to the boundary scan register during the Shift-DR state. The initialization data is transferred to the parallel outputs of the boundary scan register cells on the falling edge of TCK in the Update-DR state. The data is applied to the external output pins by the EXTEST or CLAMP instruction. System operation is not affected.

55.5.4.3 SAMPLE instruction

The SAMPLE instruction obtains a sample of the system data and control signals present at the MCU input pins and just before the boundary scan register cells at the output pins. This sampling occurs on the rising edge of TCK in the Capture-DR state when the SAMPLE instruction is active. The sampled data is viewed by shifting it through the boundary scan register to the TDO output during the Shift-DR state. There is no defined action in the Update-DR state. Both the data capture and the shift operation are transparent to system operation.

55.5.4.4 EXTEST External test instruction

EXTEST selects the boundary scan register as the shift path between TDI and TDO. It allows testing of off-chip circuitry and board-level interconnections by driving preloaded data contained in the boundary scan register onto the system output pins. Typically, the preloaded data is loaded into the boundary scan register using the SAMPLE/PRELOAD instruction before the selection of EXTEST. EXTEST asserts the internal system reset for the MCU to force a predictable internal state while performing external boundary scan operations.

55.5.4.5 ENABLE_JTAG_PASSWORD instruction

The ENABLE_JTAG_PASSWORD instruction selects the JTAG_PASSWORD register for connection as the shift path between TDI and TDO.

55.5.4.6 HIGHZ instruction

HIGHZ selects the bypass register as the shift path between TDI and TDO. While HIGHZ is active all output drivers are placed in an inactive drive state (e.g., high impedance). HIGHZ also asserts the internal system reset for the MCU to force a predictable internal state.

55.5.4.7 CLAMP instruction

CLAMP allows the state of signals driven from MCU pins to be determined from the boundary scan register while the bypass register is selected as the serial path between TDI and TDO. CLAMP enhances test efficiency by reducing the overall shift path to a

single bit (the bypass register) while conducting an EXTEST type of instruction through the boundary scan register. CLAMP also asserts the internal system reset for the MCU to force a predictable internal state.

55.5.4.8 ACCESS_AUX_x instructions

The JTAGC is configurable to allow other TAP controllers on the device to share the port with it. This is done by providing ACCESS_AUX_x instructions for each of these TAP controllers. When this instruction is loaded, control of the JTAG pins are transferred to the selected TAP controller. Any data input via TDI and TMS is passed to the selected TAP controller, and any TDO output from the selected TAP controller is sent back to the JTAGC to be output on the pins. The JTAGC regains control of the JTAG port during the UPDATE-DR state if the PAUSE-DR state was entered. Auxiliary TAP controllers are held in RUN-TEST/IDLE while they are inactive. Instructions not used to access an auxiliary TAP controller on a device are treated like the BYPASS instruction.

55.5.4.9 BYPASS instruction

BYPASS selects the bypass register, creating a single-bit shift register path between TDI and TDO. BYPASS enhances test efficiency by reducing the overall shift path when no test operation of the MCU is required. This allows more rapid movement of test data to and from other components on a board that are required to perform test functions. While the BYPASS instruction is active the system logic operates normally.

55.5.5 Boundary scan

The boundary scan technique allows signals at component boundaries to be controlled and observed through the shift-register stage associated with each pad. Each stage is part of a larger boundary scan register cell, and cells for each pad are interconnected serially to form a shift-register chain around the border of the design. The boundary scan register consists of this shift-register chain, and is connected between TDI and TDO when the EXTEST, SAMPLE, or SAMPLE/PRELOAD instructions are loaded. The shift-register chain contains a serial input and serial output, as well as clock and control signals.

55.6 Initialization/Application information

The test logic is a static logic design, and TCK can be stopped in either a high or low state without loss of data. However, the system clock is not synchronized to TCK internally. Any mixed operation using both the test logic and the system functional logic requires external synchronization.

To initialize the JTAGC block and enable access to registers, the following sequence is required:

1. Set the JCOMP signal to the JTAGC enable value, thereby enabling the JTAGC TAP controller
2. Load the appropriate instruction for the test or action to be performed

Chapter 56

IEEE 1149.7 Compact JTAG Test Access Port Controller (CJTAG)

56.1 References

- IEEE Std 1149.1-2001 (R2008), Standard Test Access Port and Boundary-Scan Architecture
- IEEE Std 1149.7, Standard for Reduced-Pin and Enhanced-Functionality Test Access Port and Boundary-Scan Architecture

56.2 Abbreviations

Table 56-1. Abbreviations

Abbreviation	Expansion
APU	Advanced Protocol Unit
APFC	Auxiliary Pin Function Control
CGM	Conditional Group Member
CGMC	Conditional Group Membership Count
CID	Controller ID
CJTAG	Compact JTAG
CLTAPC	Chip-Level TAPC
CP	Check Packet
CPA	Check Process Active
DCU	Data Channel Unit
EOT	End of Transfer
EPU	Extended Protocol Unit
Jscan	JTAG Scan
Mscan	Maximum-flexibility Scan
nTRST	Active low Test Reset
Oscan	Optimized Scan
PSS	Pause Selection State

Table continues on the next page...

Table 56-1. Abbreviations (continued)

Abbreviation	Expansion
RSU	Reset and Selection Unit
SP	Scan Packet
SPA	Scan Packet Active
Sscan	Segmented Scan
SSD	Scan Selection Directive
SSM	Scan State Machine
STL	System Test Logic
TAP	Test Access Port
TAP.1	An IEEE 1149.1 Test Access Port
TAP.7	An IEEE 1149.7 Test Access Port
TCA	Tap Controller Address
TCK	Test Clock
TCKC	Test Clock Compact
TDI	Test Data Input
TDIC	Test Data Input Compact
TDO	Test Data Output
TDOC	Test Data Output Compact
TMS	Test Mode Select
TMSC	Test Mode Select Compact
TS	Target System
ZBS	Zero Bit Scan

56.3 Introduction

The CJTAG module implements parts of the IEEE 1149.7 standard for test and debug capabilities. IEEE 1149.7 defines enhanced test and debug capabilities which are backwards compatible with IEEE P1149.1. This standard is also known as "Compact JTAG" or "CJTAG".

The module implements the following parts of the IEEE 1149.7 standard:

- IEEE 1149.1 compliant behavior while allowing multiple on-chip TAP controllers
- Ability to assert test reset
- Improved system scan performance with one-bit IR and DR chip bypass paths

- Physical connection of a Debug and Test System (external tool) to a running system without corrupting its operation
- Operation with both 4-pin and 2-pin scan topologies

56.3.1 Types of operation

The 1149.7 standard provides a scalable and flexible solution to meet the needs of varied component and system complexities. There are three types of operation:

- **Compliant**
 - Compliant with the IEEE 1149.1 standard
 - Four-pin operation using the "Standard Protocol" — IEEE 1149.1 signaling where TCK, TMS, TDI and TDO signals transfer control/data information
 - More than one TAPC may be associated with the TAP (once made visible by private instructions)
- **Extended**
 - Compatible with the IEEE 1149.1 standard
 - Four-pin operation using the "Standard Protocol" — IEEE 1149.1 signaling where TCK, TMS, TDI(C) and TDO(C) signals transfer control/data information
 - Additional test/debug functions
- **Advanced**
 - Two-pin operation using the "Advanced Protocol" — signaling where TCK(C) and TMS(C) signals transfer control/data information
 - The TMS(C) signal is operated in a bidirectional manner with the TMS, TDI and TDO TAP information serialized and transferred via this signal

56.3.2 Deployment by class

The TAP.7 capabilities are deployed using five capability classes (T0–T4), each matching the needs of certain scan topologies and mixes of 1149.1 TAPs and 1149.7 TAPs. The classes provide the following capability:

- **Compliant Operation** – provides behavior compliant with the IEEE 1149.1-2001 standard:
 - **Class T0** – 1149.1 compliance when multiple TAPs are used on the same chip.
- **Extended Operation** – extends the capabilities of the IEEE 1149.1-2001 standard:
 - **Class T1** – T0 + Test access port power management, functional reset, and test reset.
 - **Class T2** – T1 + Chip bypass (1-bit) for IR/DR Scans + hot-connection with no errors..
 - **Class T3** – T2 + Star operation (TAPs are paralleled) with series equivalent scans for test.
- **Advanced Operation** – adds capabilities addressing test and debug needs in complex environments:
 - **Class T4** – T3 + Scan with two pins with the Advanced Protocol. A number of protocol optimizations maximize scan performance. TDI(C)/TDO(C) pins are optional and may have programmable function when implemented.

56.3.3 1149.7 TAP signals

The TAP signals for the T0–T4 TAP.7 classes are listed in the following table. Signal names that end in “C” are associated with expanded functionality (beyond the functionality provided by the IEEE 1149.1 standard).

The table uses the following notation:

- M = Mandatory signal
- MC = Mandatory signal with expanded functionality
- O = Optional signal
- OC = Optional signal with expanded functionality

The CJTAG module provides T4 capability and implements the following pins:

- TCKC
- TDIC
- TDOC
- TMSC
- nTRST

Table 56-2. TAP.7 signal list

Signal name	Description	Class				
		T4	T3	T2	T1	T0
TCK/TCKC	Test clock	MC	M	M	M	M
TMS/TMSC	Test mode select	MC	M	M	M	M
TDI/TDIC	Test data input	OC	MC	M	M	M
TDO/TDOC	Test data output	OC	MC	M	M	M
nTRST	Test reset	O	O	O	O	O
Mandatory signal count (minimum)		2	4	4	4	4

56.3.4 TAP.7 architecture

The TAP.7 hardware architecture is shown in the following figure. It is described with the hardware layers listed below:

- **STL – System Test Logic** – Logic with 1149.1 compliant behavior and underlying TAP hierarchy (T0 capabilities). Provides an IEEE 1149.1 interface for the T0 TAP. 7. This logic is outside the CJTAG module.
- **RSU – Reset and Selection Unit** – A hardware layer that is placed between the APU, EPU, or STL and the TAP.7 signals (Added as an option to support the use of the Control Protocol) Provides reset and TAP.7 Controller selection services.
- **EPU – Extended Protocol Unit** – A hardware layer that is placed between the STL and the TAP.7 signals. (Added for T1, T2, and T3 capabilities). Provides an IEEE 1149.1 interface for the T1–T3 TAP.7s.
- **APU – Advanced Protocol Unit** – A hardware layer that is placed between the STL/ EPU and the TAP.7 signals (added for T4 capabilities). Provides a T4 TAP.7 interface that is either narrow or wide, with the wide version providing an IEEE 1149.1 interface.

The CJTAG module supports T4 functionality that includes STL, RSU, EPU, and APU hardware layers.

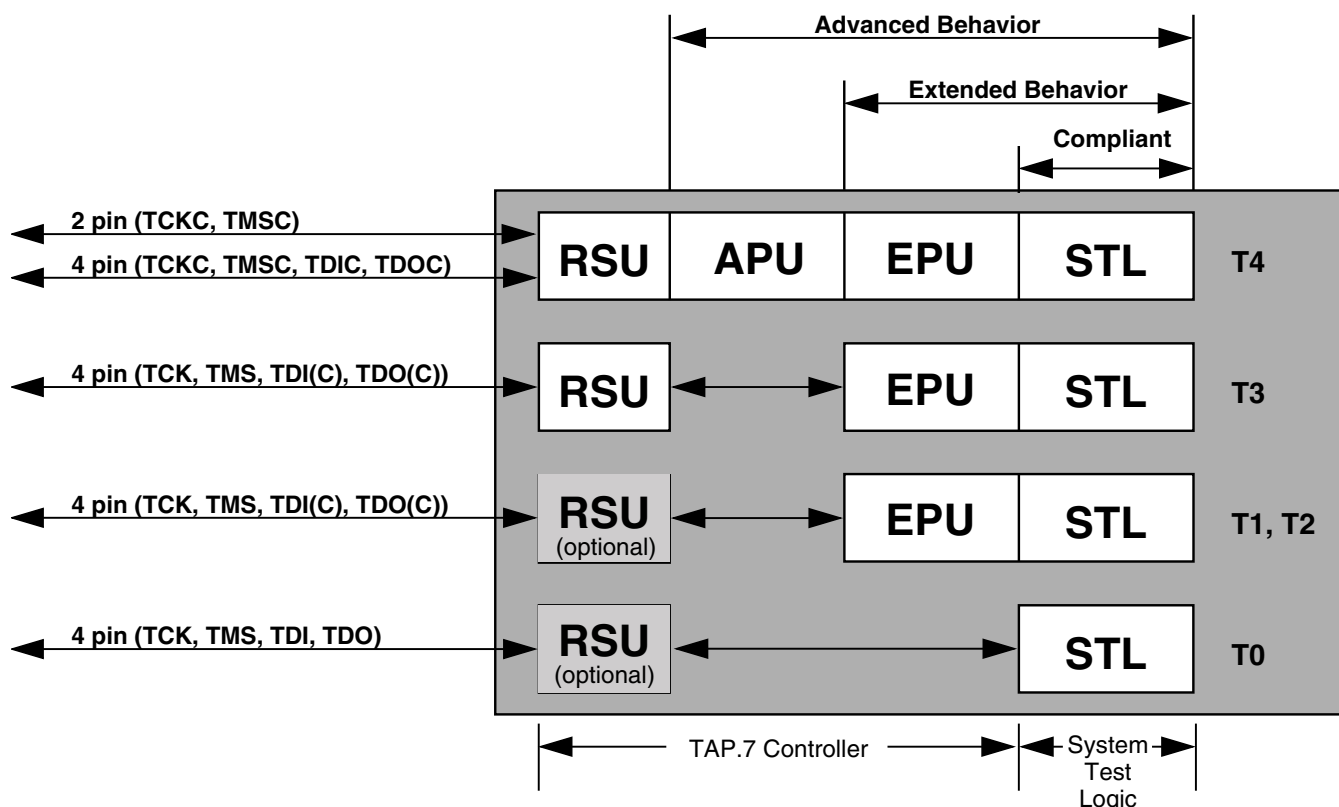


Figure 56-1. TAP.7 Controller Architecture

56.3.5 Protocols

The TAP.7 architecture utilizes the three protocols shown in the following figure.

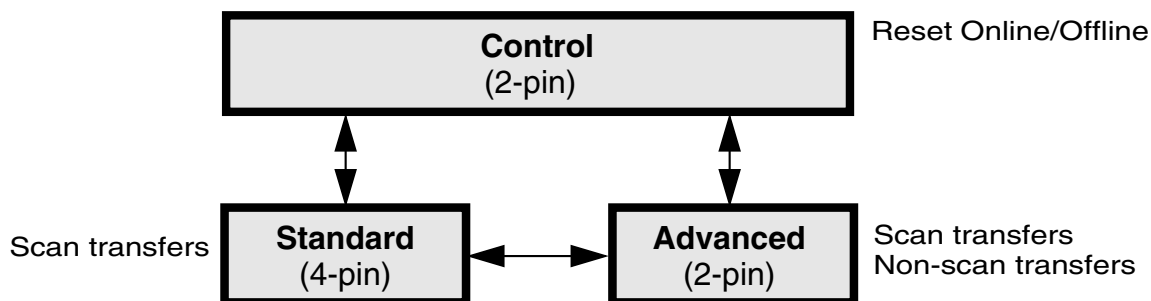


Figure 56-2. TAP.7 Behaviors

The Standard Protocol transfers data with the TMS(C) signal value being sampled with a TCK(C) edge. The Advanced and Control Protocols use both data transferred with the TMS(C) signal value being sampled with a TCK(C) signal edge and escape sequences. The Standard Protocol may be used to switch to the use of the Advanced Protocol and

vice versa. The Control Protocol can also be used to switch between the use of the Standard and the Advanced Protocol. The mandatory and optional deployment of these protocols with the TAP.7 classes is shown in the following table.

Table 56-3. Class/protocol relationship

Class	Protocol		
	Standard	Control	Advanced
T0 – T2	Mandatory	Optional	N/A
T3	Mandatory	Mandatory	N/A
T4	Mandatory	Mandatory	Mandatory

The CJTAG module implements T4 functionality and supports the Standard, Advanced, and Control Protocols.

56.3.5.1 Standard Protocol

The Standard Protocol is the signaling defined by the IEEE 1149.1 standard. This protocol is also used to implement TAP.7 commands using only the TCK(C) and TMS(C) signals in a manner that does not require either TAP.7 controller instruction or data registers. The TAP.7 controller does not add bits to the scan paths of underlying technology. The Standard Protocol can be used to manage the TAP.7 controller functionality independent of the function associated with the availability of the TDI and TDO signals. The Standard Protocol supports STL data transfers only when these signals are present and provide the 1149.1 functionality defined by the IEEE 1149.1 standard.

56.3.5.2 Advanced Protocol

The Advanced Protocol supports both STL data transfers and management of the TAP.7 controller functionality with both the 2-signal (narrow) and 4-signal (wide) configurations. The Advanced Protocol serializes the TMS, TDI, and TDO information used with the Standard Protocol using only the TCK(C) and TMS(C) signals. A number of additional scan formats provide bit sequences optimized for specific use cases. Each of these additional scan formats affects the TMSC signaling sequences.

56.3.5.3 Control Protocol

The Control Protocol uses Escape sequences to reset the TAP.7 controller, place a TAP.7 controller offline or online, and indicate End of Transmission (EOT) with the Advanced Protocol. The characteristics of this protocol provide Hot-Connect-Protection by requiring a preamble of alternating ones and zeroes preceding a selection escape when a TAP controller is started up offline.

56.4 Operating models

The following figure illustrates a simplified view of the CJTAG operating model.

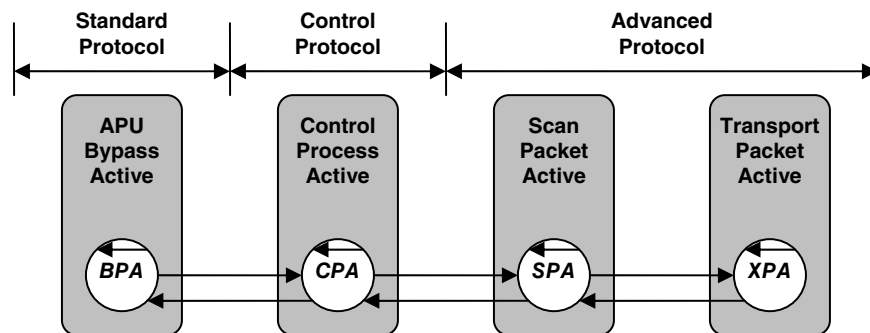


Figure 56-3. CJTAG operating model

56.5 CJTAG implementation summary

The CJTAG module is a T4 TAP.7 controller that implements the following functions.

56.5.1 T0 functions

- IDCODE. - implemented in chip level TAPC (JTAGC module)
- 1149.1 Compliance at start-up
- Isolating / Excluding of embedded TAPs – implemented in chip level TAPC (JTAGC module)
- Operation with other technologies and scan topologies
- Multiple embedded TAPs

56.5.2 T1 functions

- EPU Command processing
- EPU Class-Specific Registers
- Asserting a test reset to the STL (TRESET Register)

56.5.3 T2 functions

- “Super Bypass” function – JScan0, JScan1, JScan2 scan formats. Allows the STL to be coupled or decoupled.

56.5.4 T3 functions

- Generation of a TAP.7 controller reset with a Reset Escape Sequence
- JScan3 Scan Format

56.5.5 T4 functions

- TMSC sampling on rising edge or falling edge of TCKC
- MScan (For test and debug)
- OScan 0-7
- TDIC and TDOC / Alternate pin function multiplexing (wide 4 only)

56.6 Ancillary services

56.6.1 Overview

Ancillary services are pertinent to any TAP.7 controller implemented with an RSU, such as the CJTAG module.

Ancillary services include:

- Resets
- Start-up options

- Escape sequences
- TAPC state machine

56.6.2 Resets

The TAP.7 controller reset function expands the reset function provided by the IEEE 1149.1 standard to provide additional reset types. It supports six reset types, Type-0 – Type-5, listed below. It can be implemented with as few as two reset types (Type-2 and Type-4), or as many as six reset types (all types) of reset inputs: This is determined by the class and options implemented.

The CJTAG module utilizes Type-0, Type-2, Type-3, Type-4, and Type-5 resets.

- **Type-0** – Reset of the TAP.7 by power management logic (power-on reset)
- **Type-1** – Chip level generated Test Reset. (equivalent to Type-2)
- **Type-2** – Externally generated Test Reset (nTRST/nTRST_PD pin)
- **Type-3** – TAP.7 controller generated reset
- **Type-4** – *Test-Logic-Reset* state
- **Type-5** – TAP.7 Controller TReset Register reset of the CLTAPC.

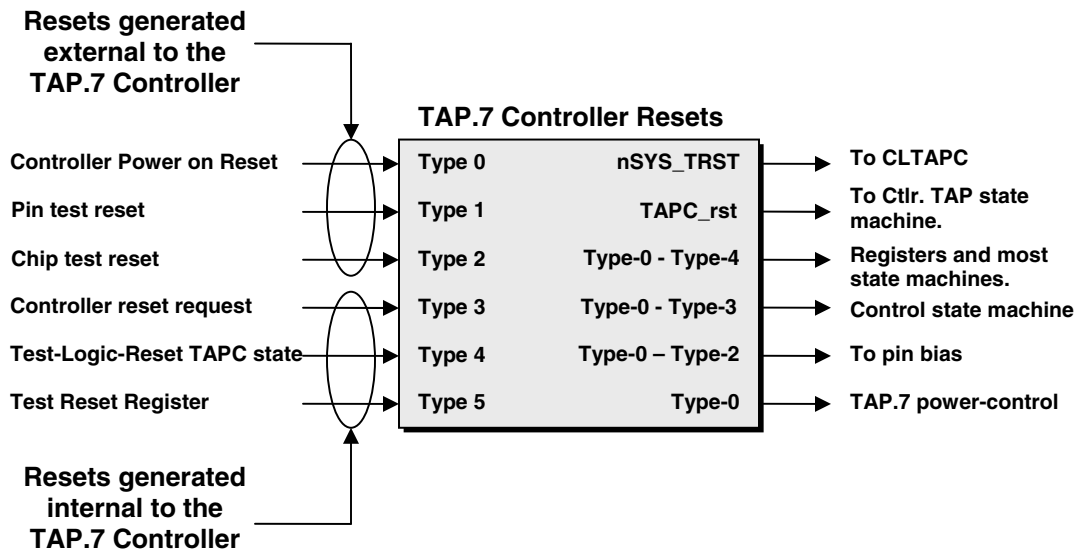


Figure 56-4. Conceptual view of the EPU reset logic and state machines

56.6.2.1 Type 5 reset

A Type-5 reset affects only the STL via the nSYS_TRST pin. It is asserted when the TRESET Register value is a logic 1. This register is set to its inactive state by reset Types 0-4.

56.6.2.2 Type 4 reset

A Type-4 reset is created by the *Test-Logic-Reset* state. It does not affect the TAP.7 TAPC state machine state as this state is the source of this reset, nor does it cause the assertion of the nSYS_TRST signal. It does however affect the coupling and decoupling of the STL. The Type-4 reset initializes the TAP.7 registers and most controller state machines. It affects but does not initialize the control state machine managing TAP.7 controller selection. It also establishes the default operating conditions.

56.6.2.3 Type 3 reset

A Type-3 reset is created by reset requests generated by the TAP.7 controller. A Type-3 reset performs the functions associated with a Type-4 reset and creates the *Test-Logic-Reset* state.

A Type-3 reset is generated in response to reset requests generated by EPU and APU logic

- A reset escape sequence
- Reset protocol sequences (delay packet directive, check packet directive)

A Type-3 reset request is generated with a reset escape sequence. This function is similar to but not the same as the reset function generated by a test reset pin. The Advanced Protocol includes certain bit patterns that initiate Type-3 resets. These bit patterns assist in the initialization of the TAP.7 controller when the TCKC signal is sourced by the TS and the external tool/TS connection is broken. The external tool may also generate these sequences, if desired, while the external tool/TS connection remains intact. These reset requests cause a Type-3 reset that is one clock wide.

With this being the case, consideration may be given to integrating the CJTAG module without the use of the Test Reset signal (nTRST/nTRST_PD pin).

56.6.2.4 Type 2 reset

A Type-2 reset is created by chip level logic to initialize the TAP.7 controller when TAP.7 power control is not implemented. It is unimplemented otherwise. It performs all of the functions of the Type-3 reset in addition to initializing the escape sequence detection logic.

56.6.2.5 Type 1 reset

A Type-1 reset is created with either the nTRST or nTRST_PD pin. It is not created when neither of these pins is implemented. It performs all of the functions of the Type-2 reset. It initiates TAP.7 controller power down provided the TAP.7 power control mode permits power down.

56.6.2.6 Type 0 reset

A Type-0 reset is implemented when TAP.7 controller power down is implemented. It performs all of the functions of a Type-1 reset and also initializes the TAP.7 power control logic.

56.6.2.7 Reset State Machine

The Reset State Machine assists in the translation of the reset inputs to the reset outputs. It handles Type-3 reset controller reset requests. The state machine state assures Type-0, Type-1, and Type-2 resets are forwarded to the STL without modifying their duration.

This state machine allows a Type-3 reset only when its state is “T3 Reset” (two TCK(C) falling edges have occurred in the absence of a Type-0, Type-1, and Type-2 reset).

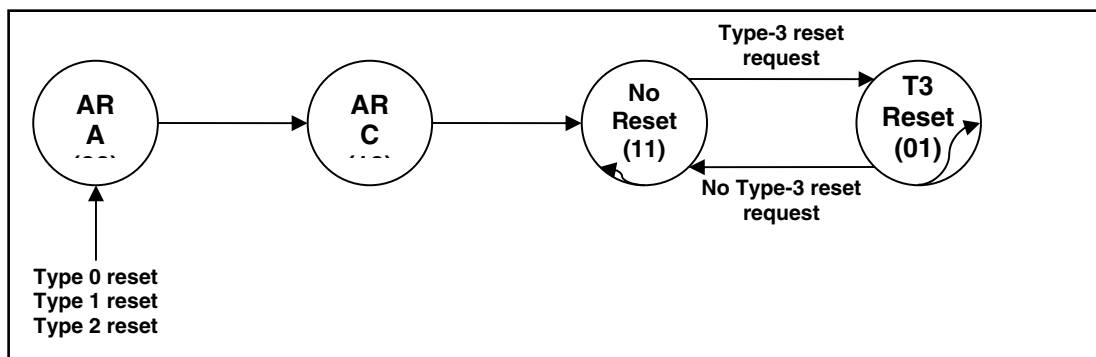


Figure 56-5. Reset state machine

Table 56-4. Reset state descriptions

Reset state		Description
ARA	Async. Reset Asserted	An async. reset has occurred or is occurring
ARC	Async. Reset Completed	An async. reset has occurred and has been released
No Reset	No Reset	No Reset
T3 Reset	Type-3 Reset	A Type-3 reset request has been processed.

Type-0, Type-1, and Type-2 resets are fully asynchronous. These resets are passed directly to nSYS_TRST without modification. The Type-3, Type-4, and Type-5 resets are fully synchronous.

The effects of the TAP.7 controller reset types are shown in the following table.

Table 56-5. Reset effects

Reset	CLTAPC	Pin-Hi-Z	State Machines other than CSM	Control State Machine (CSM)	TAPC SM State	Escape Sequence Detection	Power Control State
Type-5	Yes	–	–	–	–	–	–
Type-4	–	Yes	Yes	–	–	–	–
Type-3	Yes	Yes	Yes	Yes	Yes	–	–
Type-2	Yes	Yes	Yes	Yes	Yes	Yes	–
Type-1	Yes	Yes	Yes	Yes	Yes	Yes	–
Type-0	Yes	Yes	Yes	Yes	Yes	Yes	Yes

56.6.3 Start-up Options

56.6.3.1 Overview

The TAP.7 start-up options are shown in the following table. The only startup option currently supported by the CJTAG module is 1149.1 Compliant.

Table 56-6. Start-up Options

Start-up Option	Description
1149.1 Compliant	TAP.1 signal functionality, the CLTAPC is coupled
1149.1 Compatible	TAP.7 signal functionality, the CLTAPC is decoupled

Table continues on the next page...

Table 56-6. Start-up Options (continued)

Start-up Option	Description
1149.1 Protocol Compatible	No, or alternate TDIC/TDOC signal functionality, the CLTAPC is decoupled.
Offline-at-Start-up (Dormant)	The TAP.7 controller is offline and awaits synchronization to external tool operation so as to be placed online, the CLTAPC is decoupled after its state is moved to <i>Run-Test/Idle</i>

56.6.3.2 1149.1 Compliant start-up

Start-up behavior with the 1149.1 Compliant behavior start-up option:

- 1149.1-specific behavior / Standard protocol
- The TAPC finite state machine operated in lock step with the CLTAPC finite state machine.
- The TDIC and TDOC pins are present and provide the TDI and TDO function.
- The CLTAPC is coupled at start-up.

With this start-up option, the TAP.7 controller may be programmed and the scan topology may be interrogated. If the scan topology is known to be the Series Scan Topology at power-up, the TAP.7 can be operated as though it is a TAP.1.

Table 56-7. Reset values for TAP.7 startup options

Start-up Option	Type-0 – Type-3 Reset?	Default TAP.7 and TAP.7 controller characteristics				
		TAP.7 Controller online	CLTAPC Coupled	SGC Reg. == 1	SCNFMT	Initial TMSC pin bias
1149.1 Compliant	x	Yes	Yes	Yes	JScan0	PU

56.6.4 RSU operation

The RSU uses TCKC and TMSC with the control protocol for configuration control.

56.6.4.1 General Operation

From a very high level, the RSU enables the remainder of the TAP.7 functionality when the TAP.7 controller is Online and disables the remainder of the TAP.7 functionality otherwise. The RSU either consumes or discards the TMS(C) information while the TAP.7 controller is Offline. The relationship of the escape sequence detection, technology selection logic, and the underlying technology is shown in the following figure.

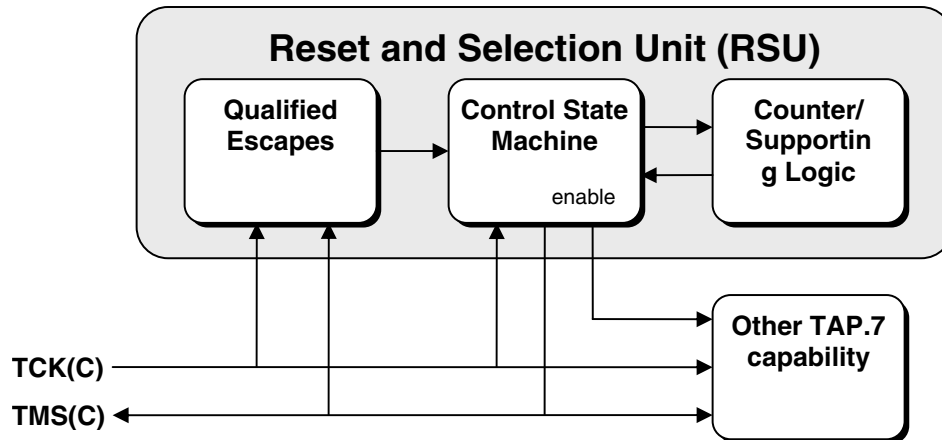


Figure 56-6. Conceptual view of technology selection mechanism

56.6.4.2 Common Signaling Across Technologies

With the TAP.7 architecture, the signaling conventions chosen for technology selection is multiple data transitions of the TMS(C) signal while the TCK(C) signal is a logic 1. The common signaling space is available when the external tool sources the TCK(C) signal as it is capable of creating these signaling conventions. It is unavailable when the TS sources the TCK(C) signal as the external tool cannot stop the TCKC signal at a logic 1 to create these signaling conventions.

56.6.4.3 Escape Sequence Detection

TAP.7 TAPC interprets the count of the number of TMS(C) edges while TCK(C) is a logic 1 as one of four escape sequences. Each escape sequence has a different function and TMSC edge count. These escape sequences, their edge counts, and their functions are described below:

- **Custom** (2 or 3 edges) – Ends scan and transport transfers with a TAP.7 controller, may be used for other purposes with another technology.
- **Reset** (> 7 edges) – Resets all technologies (generates a Type-3 TAP.7 TAPC reset)

The external tool creates an escape sequence by generating one or more TMS(C) edge pairs while the TCK(C) signal is a logic 1 value. Even and odd TMS(C) edge counts beginning with two are given the same meaning.

An escape sequence:

- Can be detected only when a Type-0, Type-1 or Type-2 reset is **not** asserted.
- Begins and ends while the TCK(C) is a logic 1.
- Uses TMS(C) as a clock while the TCK(C) is a logic 1.
- Overlays additional control information onto the normal information that is transferred with the TCK(C) and the TMS(C) without changing the normal information.
- Must be detected when TCK(C) is a logic 1 regardless of the TCK(C) and TMS(C) drive histories.

56.6.4.3.1 Custom escape sequence

An online technology can use the custom escape sequence in any manner it chooses. It is used as the End of Transmission (EOT) of certain data types (formats OScan4-7) with T4 and above TAP.7s.

56.6.4.3.2 Reset escape sequences

A reset event initializes all technologies sharing the TCK(C) and TMS(C) connectivity. A reset escape sequence resets the technology. It is recommended that this reset occur with the falling edge of TCKC. With a TAP.7 controller a reset escape sequence generates a Type-3 reset beginning with the falling edge of TCK(C) following the asynchronous detection of the event.

56.6.5 TAPC State Machine

T1 and above TAP.7s require knowledge of the TAPC state machine state. With T2 and above TAPs this knowledge must be developed independently of the CLTAPC. This is generally accomplished by implementing a TAPC state machine within the EPU. This TAPC state machine provides the function of the state machine within a standard 1149.1 TAPC. It is recommended that this state machine is implemented to change state on the falling edge of TCK(C) as this has numerous advantages to a TAP.7 controller. The EPU does not contain 1149.1 Instruction or Data Registers. The APU controls the state changes of this TAPC state machine with T4 and above TAP.7s.

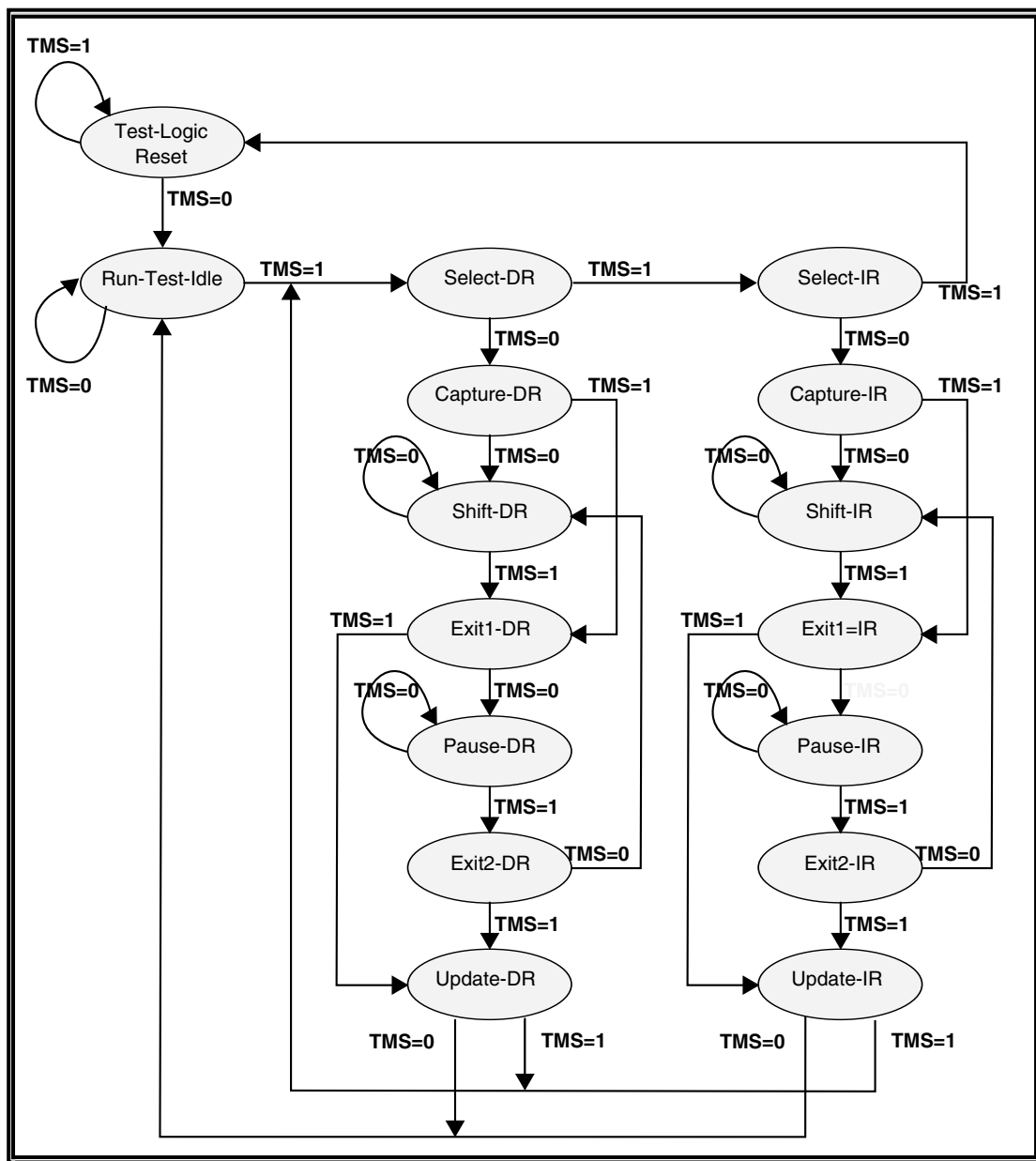


Figure 56-7. Falling edge TAP controller state machine

56.7 EPU (Extended Protocol Unit) Operation

56.7.1 EPU Operation

The EPU adds functionality to the BYPASS and IDCODE instructions.

56.7.1.1 Zero-bit DR-Scans (ZBS)

A zero-bit DR-Scan (ZBS) is a TAPC-state sequence that begins with the *Select-DR-Scan* state and ends with an exit from the *Update-DR* state without an intervening *Shift-DR* state. The external tool controls the EPU using ZBSs.

The use of ZBSs can begin immediately following the *Test-Logic-Reset* state since this state initializes the Instruction Registers of TAPs with either the *BYPASS* or *IDCODE* instructions.

56.7.1.2 Utilizing ZBSs for TAP.7 Controller Functionality

Beginning with a ZBS count of zero, the ZBS count is incremented with each consecutive occurrence of a ZBS without encountering a *Shift-DR* state. This count cannot be incremented past seven. *Run-Test/Idle* is the only state that may occur between consecutive occurrences of ZBSs without terminating the count of consecutive ZBSs.

56.7.1.3 Locking the ZBS Count

When a DR-Scan containing a *Shift-DR* state occurs, and the ZBS count is greater than zero, the ZBS count is not incremented when the *Update-DR* state is reached. It is instead locked at its current value. ZBSs occurring after locking the ZBS count do not affect the locked ZBS count. Locking the ZBS count is the equivalent of storing the count for subsequent use.

56.7.1.4 Control levels

Locking the ZBS count activates a control level that is equal to the locked ZBS count (1-7) when ZBSs are being used for TAP.7 functionality and merely locks the ZBS count otherwise.

1. ZBSs may be used by the STL or another function
2. TAP.7 command generation
3. Reserved control level
4. Accesses to optional TAP.7 scan paths
5. Accesses to optional TAP.7 scan paths

6. Force offline operation (optional)
7. External tool uses

Control level one may be used by the STL since there is no TAP.7 controller functionality associated with this control level.

Control level two enables DR-Scans to create TAP.7 commands. A portfolio of these commands supports the deployment of TAP.7 controllers in the Series or Star Scan Topologies. These commands manage TAP.7 controller registers and perform other functions.

Control level three is reserved for use with a subsequent specification revision.

Control levels four and five provide access to custom (private) scan paths when access to these paths is permitted by a TAP.7 register bit used for this purpose. The EPU provides a 1-bit scan path otherwise. No other EPU functionality is associated with these control levels. The CJTAG module does not support the use of this control level.

Control level six may be optionally used to force offline operation with a TAP.7 controller that supports this capability. The EPU provides a 1-bit scan path for this control level. The CJTAG module does not support the use of this control level.

Control level seven is reserved for external tool use. With this control level, external tool capabilities may be managed using the same infrastructure used to manage TAP.7 controller capability. The EPU provides a 1-bit scan path for this control levels.

56.7.1.5 Exiting a Control Level

A control level is exited when one of the following occurs:

- The *Select-IR-Scan* TAP.7 controller state
- The *Test-Logic-Reset* state
- Certain TAP.7 controller commands (Exit Control Level – ECL register bit)
- An event that synchronizes the operation of TAP.7 controllers

56.7.1.6 Zeroing the ZBS Count

Events causing an exit from a control level also zero the ZBS count. Scan Selection Directives (SSDs) associated with T3 and above TAP.7s zero the ZBS count when the control level has not been locked.

56.7.1.7 Utilizing ZBSs within the STL

After the *Test-Logic-Reset* state, ZBSs are used for basic EPU control. ZBSs that are intended for TAP.7 use are henceforth called "**TAP.7 ZBSs**". ZBSs that are intended for other uses are henceforth called "**Other ZBSs**". Two TAP.7 registers enable the STL's use of ZBSs. Storing these registers causes a control level exit, zeroes the ZBS count, and enables the use of Other ZBSs. Storing the Suspend Register (**SUSPEND**) suspends the use of control levels. Storing the ZBS Inhibit Register (**ZBSINH**) inhibits ZBS detection.

When either of these methods is used, the *Test-Logic-Reset* state enables the use of TAP.7 ZBSs in addition to initializing the TAP.7 controller.

The operation of the TAP.7 controller changes with the use of these two paths.

When the System Path is used:

- A count of eight consecutive ZBSs is used to create a logic 0 SUSPEND Register value.
- The CLTAPC supplies the TAP.7 scan path and controls the drive of TDO(C), provided it is not bypasses.
- The EPU supplies the TAP.7 scan path (a one bit bypass) when the CLTAPC is bypassed.
- Functions supporting Boundary Scan in a Star Scan Topology are enabled.

When the EPU Path is used:

- When the control level has not been locked, the TDO(C) signal remains high-impedance.
- The command control level may be created
- When commands are used, the TDO(C) signal remains high-impedance except during *Shift-DR* states within CR-Scans, and remains high impedance at other times.
- Auxiliary scan paths may be accessed with control levels four and five.

56.7.2 EPU Registers

56.7.2.1 Global

A global register is a register whose value is stored at the same time and with the same value as registers with the same name in all TAP.7 TAPCs sharing an external tool connection. They are stored using TAP.7 TAPC commands dedicated for this purpose. These commands store a single global register. Global registers manage TAP.7 TAPC functions that affect the synchronized operation of a TAP.7 TAPC. An example of a global register is the Scan Format (SCN_FMT) Register used to define the TAPC signal protocol.

A second means of storing all global registers is available. The selection of the TAP.7 TAPC with the technology selection function stores either the Scan Format register value (short form) or all global registers (long form). Storing all global registers provides a simple means to synchronize the values of all global registers when Technology Selection causes a non-operating TAP.7 TAPC to become operational while other TAPCs are operational.

56.7.2.2 Local

Local registers manage TAP.7 TAPC functions that do not affect the synchronized operation of TAP.7 TAPCs. Local registers are conditionally stored using TAP.7 controller commands dedicated for this purpose. These commands provide for changing the value of these registers one at a time with a unique value in a Series Scan Topology based on the TAPC's position on the scan chain. Local registers are stored with a unique value in a Star Scan Topology based on a TAPC address assigned to the TAPC.

Registers programmed with commands are described in this section. The CJTAG module implements all mandatory T4 registers as well as the optional APFC, TRESET, and RDBACK0 registers. There are 16 mandatory and 10 optional registers that can be grouped in classes as shown in Read-only – Configuration and register read back..

- **Control** – TAP.7 controller behaviors and characteristics.
- **Options** – Control of optional functions.
- **Select** – Controls the selection of a TAP.7 for Scan and execution of conditional commands.
- **Read-only** – Configuration and register read back.

Table 56-8. TAP.7 controller register list (managed with commands)

Register type	Global/local	Width	Register mnemonic	Name
Control	Global	1	ECL	Exit Control Level
		1	SUSPEND	Suspend
		1	ZBSINH	ZBS Detect Inhibit
		5	SCNFMT	Scan Format
		1	SSDE	SSD Enable
		2	DLYC	Delay control
		2	RDYC	Ready Control
Options	Local	1	TRESET	Test Reset
		2	APFC	Aux. Pin Func. Cntl.
Select	Local	1	CGM	Cond. Group Member
		1	SGC	Scan Group Candidate
		1	SEDGE	Sampling Edge
Read-only	Local	32	RDBACK0	Read-back 0
		32	RDBACK1	Read-back 1
		32	CNFG0	Configuration Register 0

M = Mandatory register

O = Optional register

56.7.2.3 Register Descriptions

56.7.2.3.1 Exit Control Level (ECL)

Storing this register zeroes the ZBS count and unlocks the control level.

This register is set using the STMC command. It remains logic 1 only momentarily before immediately returning to logic 0.

This register is cleared by a global register load.

56.7.2.3.2 Suspend (SUSPEND)

Setting this register suspends the use of control levels. When this register is a logic 1, the "System Path" (STL) is used.

1 – Suspend use of control levels. Enables "System Path."

0 – Enable use of control levels.

SUSPEND is set with the STMC command.

This register is cleared when 8 consecutive ZBSs are detected without the locking of the ZBS count.

This register is cleared by a global register load.

56.7.2.3.3 ZBS Inhibit (ZBSINH)

Storing to ZBSINH inhibits ZBS detection.

1 – Inhibit ZBS detection.

0 – Enable ZBS detection.

ZBSINH is set with the STMC command.

This register is cleared by a global register load.

56.7.2.3.4 Scan Format (SCNFMT[4:0])

Set using Store Scan Format (STFMT) command.

Read using Scan String (SCNS) command.

Table 56-9. Scan Format Register Values

TAPC.7 Class	SCNFMT	Description
T2–T4	00000	JScan0: • 1149.1 Compliant Operation. • Requests the coupling of the CLTAPC when the Run-Test/Idle state is reached.
	00001	JScan1: • 1149.1 Compatible Operation • Requests the decoupling of the CLTAPC when the Run-Test/Idle state is reached.
	00010	JScan2: • 1149.1 Compatible Operation. • Requests the coupling or decoupling of the CLTAPC when the Run-Test/Idle state is reached based on the value of the CACT register

Table continues on the next page...

Table 56-9. Scan Format Register Values (continued)

TAPC.7 Class	SCNFMT	Description
T3–T4	00011	JScan3: • Non-compatible 1149.1 Operation • Allows the detection of a T2 TAP.7 controller that is deployed in a Star-4 topology. • Enables TDOC signal behavior that prevents drive conflicts in a Star-4 Scan Topology. • Enables the use of SSDs
T4	01000	OScan0 • Data rate dependent components • Single TAP.7 communication • Debug and Test applications
	01001	OScan1 • TAP.1 components • Single/Multi-TAP.7 communication • Debug and Test applications
	01010	OScan2 • TAP.1 components • No TDI or TDO in non-shift states • Debug applications
	01011	OScan3 • TAP.1 components • No TDI in non-shift states • No TDO in any state

Table continues on the next page...

Table 56-9. Scan Format Register Values (continued)

TAPC.7 Class	SCNFMT	Description
T4	01100	OScan4 - Data rate dependent components - Single TAP.7 communication - Debug and Test applications
	01101	OScan5 - TAP.1 components - Single/Multi-TAP.7 communication - Debug and Test applications
	01110	OScan6 - TAP.1 components - No TDI or TDO in non-shift states - Debug applications
	01111	OScan7 - TAP.1 components - No TDI in non-shift states - No TDO in any state
	10000	MScan - Data rate dependent components - Multi TAP.7 communication - Directed CID assignment - Virtually any IP with compliant or non-compliant 1149.1 behavior
	10001 – 11111	Reserved

The use of a Scan Format value other than 00000b – 10000 shall not change the value of the SCNFMT Register.

56.7.2.3.5 Scan Selection Directive Enable (SSDE)

This register specifies whether the Scan Selection State may be managed using the scan selection directives.

0 – Scan selection directives are not enabled.

1 – Scan selection directives are enabled.

This register is set using the STMC command.

This register is read using SCNS (Scan String), RDBACK0.

56.7.2.3.6 Delay Control (DLYC[1:0])

The Delay Control register provides a means for the external tool to delay the completion of an SP. This delay is essentially an external tool stall. This delay may be zero, one, two, or n TCKC periods. This delay is especially useful when the TCKC is sourced by the TS and a Standard-to-Advanced Protocol adapter is added to an external tool supporting only the Standard Protocol. It allows an external tool design additional time to receive TDO, advance the external tool TAPC state, receive TMS and TDI information, and begin generation of a new SP payload. The DLYC register should be programmed before the use of the Advanced Protocol begins after the use of the Standard Protocol.

Set using the STMC command.

Read using SCNS, RDBACK0

Bits 18-17 of the "Global Register."

56.7.2.3.7 Ready Control (RDYC[1:0])

The Ready Control register defines behavior exhibited in the SP payload output bit frames when the STL stall opportunities (RDY bits) are included as control information. This register defines the number of additional bits inserted in these bit frames. These bits provide more time for a high-performance external tool to ascertain whether the TAP.7 controller is ready to complete the SP payload. The use of these bits makes a high-performance external tool design easier as input and output buffer delays play less of a role in limiting the TAP.7 controller performance. The RDYC register should be programmed before the use of the Advanced Protocol begins.

Set using the STMC command.

Read using SCNS, RDBACK0.

Bits 18-17 of the Global register state.

56.7.2.3.8 Test Reset (TRESET)

Asserts a test reset to the STL.

0 – The nSYS_TRST and SYS_TMS signals presented to the STL is not influenced by this bit.

1 – The nSYS_TRST presented to the STL is asserted and the SYS_TMS signal is a logic 1.

Set using the STC1 command.

The effects of the TRESET Register may be modified by a private TAP.7 controller register. When this private register is unimplemented or has a value of zero, the TRESET register causes the maximum initialization of the STL with nSYS_TRST and the SYS_TMS logic 1 value. Otherwise, the value of the private register may modify the function of the TRESET Register bit.

When the TRESET Register is a logic 1, the STL scan path appears broken as the state of CLTAPC remains *Test-Logic-Reset* while the EPUTAPC state progression continues. When the operation of the EPUTAPC and CLTAPC controllers is coupled, as with a T1 TAP.7, the state of these two TAPCs may be resynchronized to the *Run-Test/Idle* state by:

- An STC1 Command that sets the TRESET Register value to a logic 0.
- A stay in the Run-Test/Idle state of two or more SYS_TCK periods immediately following the Update-DR state of this command

A failure to follow the guidelines is considered a programming error and will result in erroneous system operation. The EPUTAPC and CLTAPC states will not be synchronized as they progress through the state diagram.

56.7.2.3.9 Auxiliary Pin Function Control (APFC[1:0])

This register enables TDI and TDO alternate functions when 2-pin TAP.7 operation is selected.

0x – TDI and TDO have JTAG function.

1x – TDI and TDO have alternate function.

Set using STC2 command.

Read using SCNS, RDBACK0.

56.7.2.3.10 Conditional Group Member (CGM)

Used by STC1, STC2, STTESTM, and EXC3 – EXC0 commands as the “condition” for updating registers.

Set and cleared using MCM and SCNB commands.

Read using SCNS, RDBACK0.

56.7.2.3.11 Scan Group Candidate (SGC)

This register determines whether the STL is coupled when the *Run-Test/Idle* state is reached when scan formats other than JScan0 and JScan1 are used.

- **0** – Decoupling action.
- **1** – Coupling action.

Set and cleared using STMC, MSC, and SCNB commands.

Read using SCNS, RDBACK0.

56.7.2.3.12 Sampling Edge (SEDGE)

The Sampling Edge register defines the TCKC edge used to sample the TMSC input.

- **0** – Sample TMSC on the TCKC falling edge.
- **1** – Sample TMSC on the TCKC rising edge.

Set and cleared using STC1 command.

Read using SCNS, RDBACK0

56.7.2.3.13 Read Back 0 (RDBACK0)

Non-transport related register information.

Read using SCNS command.

Unimplemented registers shall read back as "0".

Table 56-10. RDBACK register format

Bit	Width	Register Mnemonic	Name
4:0	5	SCNFMT	Scan Format
6:5	2	PWRMODE	Power-control Modes
7	1	FRESET	Functional Reset
8	1	TRESET	Test Reset
9	1	SGC	Scan Group Candidate
		CGM	Conditional Group Member
11	1	SSDE	Scan Selection Directive Enable
13:12	2	TOPOL	Topology
14	1	SREDGE	Sampling Edge
16:15	2	DLYC	Delay Control
18:17	2	RDYC	Ready Control
20:19	2	APFC	Auxiliary Pin Functional Control

Table continues on the next page...

Table 56-10. RDBACK register format (continued)

Bit	Width	Register Mnemonic	Name
22:21	2	STCKDC	SYS_TCK Duty Cycle
31:23	11	Reserved	Read as a logic 0

56.7.2.3.14 Configuration Register 0 (CNFG0)**Table 56-11. Configuration Register 0 Format**

Field	Bit #	Width	Name	Description
CNFG (Class)	3:0	4	CLASS[3:0]	TAP.7 Class
				0000 – TAP.1 or TAP.7 Class T0
				0001 – Class T1
				0010 – Class T2
				0011 – Class T3
				0100 – Class T4
				0101 – Class T5
CNFG (Revision)	7:4	4	REV[3:0]	TAP.7 specification revision.
				0000 – Reserved
				0001 – IEEE Std 1149.7-2008
				0010 – 1111 - Reserved
CNFG (Format)	11:8	4	CNFGFMT[3:0]	Configuration Register Format
				xxx1 – CNFG0[31:12] are implemented
				xx1x – CNFG1 implemented
				x1xx – CNFG2 implemented
				1xxx – CNFG3 implemented

Table continues on the next page...

Table 56-11. Configuration Register 0 Format (continued)

Field	Bit #	Width	Name	Description
T1 Options	12	1	RDBKS	RDBACK supported
				0 – Unsupported
				1 – Supported
	13	1	TRESETS	TRESET supported
				0 – Unsupported
				1 – Supported
	14	1	FRESETS	TRESET supported
				0 – Unsupported
				1 – Supported
	17:15	3	PWRMODES[2:0]	PWRMODES[x] – Power Control Mode
				0 – Unsupported
				1 – Supported
	18	1	PCMR	PCMR – PM default POWRMODE value
				0 – No
				1 – Yes
T2 Options	19	1	DCAS	DCAS – Decouple at start-up
				0 – No
				1 - Yes

Table continues on the next page...

Table 56-11. Configuration Register 0 Format (continued)

Field	Bit #	Width	Name	Description
T4 Options	21:20	2	SSCANS[1:0]	x0 – SScan1:0 Unsupported
				x1 – SScan1:0 Supported
				0x – SScan3:2 Unsupported
				1x – SScan3:2 Supported
	24:22	3	OSCANS[2:0]	xx0 – OScan3:2 Unsupported
				xx1 – OScan3:2 Supported
				x0x – OScan5:4 Unsupported
				x1x – OScan5:4 Supported
				0xx – OScan7:6 Unsupported
				1xx – OScan7:6 Supported
	25	1	APFCS	Auxiliary Pin Function Control supported
				0 – Unsupported
				1 – Supported
	26	1	TAPWIDS	TAP Width Default
				0 – 2-pin supported
				1 – 4-pin supported
	27	1	TAPWLCK	TAP Width Locked
				0 – Not locked
				1 - Locked
	28	1	STCKDCS	SYS_TCK Duty Cycle supported
				0 – Unsupported
				1 – Supported
Reserved	31:29	3	Reserved	Reserved, read as zero

56.7.2.3.15 Register reset values

Table 56-12. Reset values of TAP.7 controller registers managed with commands

Register type	Width	Register mnemonic	Name	Values for reset type			
				0	1 – 3	4	5
Control	1	ECL	Exit Control Level	0	0	0	NC
	1	SUSPEND	Suspend	0	0	0	
	1	ZBSINH	ZBS Detect Inhibit	0	0	0	
	5	SCNFMT	Scan Format	DF	DF	DF	
	1	SSDE	SSD Enable	0	0	0	
	2	DLYC	Delay control	00	00	00	
	2	RDYC	Ready Control	00	00	00	
Options	1	TRESET	Test Reset	0	0	0	NC
	2	APFC	Aux. Pin Functional Cntl.	DP	DP	NC	
Select	1	CGM	Conditional Group Member	0	0	0	NC
	1	SGC	Scan Group Candidate	DCA	DCA	DCA	
	1	SREDGE	Sampling Rising Edge	0	0	0	
Read-only	32	RDBACK0	Read-back 0	N/A	N/A	N/A	N/A
	32	CNFG0	Configuration Register 0				

NC = No Change

DF = Default Scan Format

DM = Default Power Control Mode

DCA = Default Coupling Action

DT = Default Topology

DP = Default Pin Function

56.7.3 EPU Commands

There are two command types:

- Two Part Commands
- Three Part Commands

Any number of two-part and three part commands may be issued in any combination before the command control level is exited. If the command control level is exited before a two- or three-part command is completed, the command is aborted and becomes a NOP.

56.7.3.1 Two Part Commands

TAP.7 controller commands are 10-bit values. The 10-bit commands are created by two consecutive DR-Scans when the control level is locked at the command control level (two). The first DR-Scan provides a five-bit command opcode. The second DR-Scan provides a five-bit operand. These two scans are called the command part one (CP1) and command part two (CP2), respectively. A command created entirely with these two scans is called a two-part command.

56.7.3.2 Three Part Commands

Some commands are used to send and/or receive data values other than values embedded in the command's operand. These commands are called three-part commands. With these commands, CP1 and CP2 are followed by an additional DR-Scan to transport a data value to or from an EPU scan path. When used for this purpose, the DR-Scan is called a Control Register Scan (CR-Scan). The CR-Scan is a minimum of zero bits in length with there being no maximum length. The CR-Scan path is described in SECTION.

56.7.3.3 Command Sequence

The two 5-bit values concatenated to create the 10-bit command represent the number of Shift-DR states between the *Capture-DR* and *Update-DR* states of CP1 and CP2. The count within CP1 creates the MSBs of the command while the count within CP2 creates the LSBs of the command. The command is decoded when the *Update-DR* state of the second DR-Scan is reached. A determination as to whether the command is a two- or three-part command is made at this point. The command format is shown in the following figure.

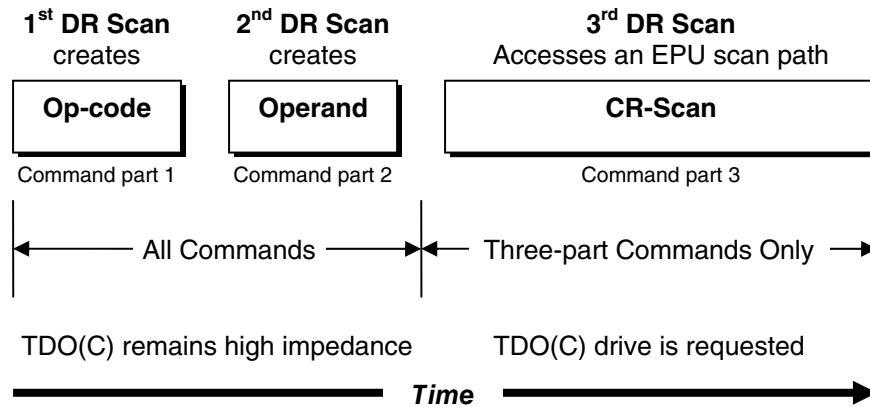


Figure 56-8. DR_Scan sequence used for command creation

The 10-bit command created by CP1 and CP2 determines the command's function and whether the command is a two- or three-part command. If the command is a two part command, the function specified by the command is performed when command part 2 completes. If the command is a three-part command, the DR-Scan following CP2 performs a TAP.7 controller function designated by the 10-bit controller command. This DR-Scan is called a **Control-Register Scan** (CR-Scan) and can be any length. A CR-Scan has many of the attributes of the DR-Scan as it moves data between the external tool and TAP.7 controller and accesses various EPU scan paths. There are many two part commands, but only three three-part commands, each having a special purpose.

56.7.3.4 Commands

There are 11 mandatory and 5 optional commands that can be grouped in classes as shown in the following list:

- Store – The operand is stored in a register or causes an action.
- Select – Controls participation of Scan and conditional command execution.
- Scan – Inputs and outputs with a CR-Scan.
- Enumerate – Controller ID allocation and de-allocation.
- Private – Commands available for chip-specific definition.

The definition of the commands with their operands is shown in the following table.

Table 56-13. – TAP.7 controller command list

Command class	Opcode count	Mnemonic	Name	TAP.7 class			
				1	2	3	4
Store	0x00	STMC	Store Miscellaneous Control	M	M	M	M
	0x01	STC1	Store Conditional 1-bit	M	M	M	M
	0x02	STC2	Store Conditional 2-bit	M	M	M	M
	0x03	STFMT	Store Format	-	M	M	M
	0x04	STDCST	Store Data Channel State	-	-	-	-
Select	0x06	MCM	Make Cond. Group Member	M	M	M	M
	0x07	MSC	Make Scan Group Candidate	-	M	M	M
Scan	0x08	SCNB	Scan Bit	M	M	M	M
	0x09	SCNS	Scan String	M	M	M	M
Enumerate ¹	0x0A	CIDA	Allocate Controller ID	-	-	M	M
	0x0B	CIDD	De-allocate Controller ID	-	-	M	M
Reserved	0x0C–0x1F	Reserved	Reserved	-	-	-	-

1. Enumerate commands are not currently supported by the CJTAG module.

56.7.3.4.1 Store Commands

Store commands are two-part commands that are available in both Star and Series Scan Topologies. The Store Miscellaneous Control (STMC) Command performs control functions that affect all TAP.7 controllers sharing an external tool connection. It stores values in one-bit and two-bit registers.

The Store Conditional 1 bit (STC1) and Store Conditional 2 bit (STC2) Commands store one-bit and two-bit values in registers, respectively. These commands operate two ways:

1. Unconditionally storing registers in all TAP.7 controllers or
2. Storing registers in only the TAP.7 controllers that have a logic 1 Conditional Group Member (CGM) Register value.

The Store Format (STFMT) Command unconditionally stores a five-bit Scan Format value.

The Store Data Channel State (STDCST) Command stores a five-bit value identifying the states supporting transport.

56.7.3.4.2 Select Commands

Select commands are two-part commands that are available in both Star and Series Scan Topologies. Their primary use is to select TAP.7 controllers of interest in a Star Scan Topology. These commands define the TAP.7 controller(s) participating in scans and the TAP.7 controllers that conditionally execute STC1 and STC2 Commands.

The Make Conditional Group Member (MCM) Command specifies which TAP.7 controllers execute store conditional commands as it manages the Command and Path Enable (CAPE) Register. This register also affects the execution of private commands and is utilized in EPU scan path selection.

The Make Scan Group Candidate (MSC) command manages the SGC register. This register determines whether the STL is coupled when the *Run-Test/Idle* state is reached when scan formats other than JScan0 and JScan1 are used. Select commands execute only when the CIDI Register is a logic 0, indicating the TAP.7 controller's CID is valid, and are treated as no operations otherwise.

56.7.3.4.3 Scan Commands

Scan commands are three-part commands that are available in both Star and Series Scan Topologies. The CR-Scan portion of Scan Bit (SCNB) and Scan String (SCNS) Commands is used to set and test bits (SCNB) or transfer strings of bits (SCNS). The commands may be used to access both public and private registers.

56.7.3.5 Command Definitions

56.7.3.5.1 Store Commands

56.7.3.5.1.1 Store Miscellaneous Control Command (STMC)**Table 56-14. Store Miscellaneous Control**

STMC		Store Miscellaneous Control				
Opcode	Operand	bbb		x	y	Description
00000	bbb x y	000	StateCtl	0	0	NOP
				0	1	ExitCmdLev(ECL)
				1	0	Exit/Suspend (SUSPEND) = 1
				1	1	ZBS Inhibit (ZBSINH) = 1
		001	ScanCtl	0	-	SGC = y
				1	-	CGM = y
		010	RdyCtl	-	-	RDYC = xy
		011	DlyCtl	-	-	DLYC = xy
		100	ScnFunc	0	-	SSDE = y
				1	0	Star-4 Topology Test
				1	1	Reserved
		101	Reserved	-	-	Reserved
		110				
		111				

56.7.3.5.1.2 Store Conditional 1-bit (STC1)**Table 56-15. Store Conditional 1 bit (STC1)**

STC1		Store Conditional 1 bit			
Opcode	Operand	c	bbb	CAPE	Description
00001	c bbb v	0	000	-	SREDGE = v
			001	-	Reserved
			010	-	TRESET = v
			011 – 111	-	Reserved
		1	—	0	No change
			000	1	SREDGE = v
			001	1	Reserved
			010	1	TRESET = v
			011 – 111	1	Reserved

56.7.3.5.1.3 Store Conditional 2-bit (STC2)**Table 56-16. Store Conditional 2 bit (STC2)**

STC2		Store Conditional 2 bit			
Opcode	Operand	c	bb	CAPE	Description
00010	c bb vv	0	00	-	Reserved
			01	-	STCKDC = vv
			10	-	APFC = vv
			11	-	Reserved
		1	—	0	No change
			00	1	Reserved
			01	1	STCKDC = vv
			10	1	APFC = vv
			11	1	Reserved

56.7.3.5.1.4 Store Scan Format (STFMT)**Table 56-17. Store Scan Format (STFMT)**

STFMT		Store Scan Format
Opcode	Operand	
00011	nnnnn	SCNFMT = nnnnn

56.7.3.5.2 Select Commands

This section describes the select commands with their operands.

56.7.3.5.2.1 Make Conditional Group Member (MCM)

This command operates as a no operation when the CIDI Register is a logic 1 (The Controllers CID is invalid). Otherwise it sets the CGM bit in a TAP.7 controller that has a valid CID and the CID matches the operand CID field (iiii). The CGM Register is also stored with the STMC and SCNB commands.

Table 56-18. Make Conditional Group Member (MCM)

MCM		Conditional Group Member	
Opcode	Operand	m	Description
00110	m iiii	0	CGM bit of the non-targeted controller is cleared
		1	CGM bit of the targeted controller is set. CGM bit of a non-targeted controller is unaffected.

56.7.3.5.2.2 Make Scan Group Candidate (MSC)

This command operates as a no operation when the CIDI Register is a logic 1 (The Controllers CID is invalid). Otherwise it sets the SGC bit in a TAP.7 controller that has a valid CID and the CID matches the operand CID field (iiii). The SGC Register is also stored with the STMC and SCNB commands.

Table 56-19. Make Scan Group Candidate (MSC)

MSC		Make Scan Group Candidate	
Opcode	Operand	m	Description
00111	m iiii	0	SGC bit of the non-targeted controller is cleared
		1	SGC bit of the targeted controller is set. SGC bit of a non-targeted controller is unaffected.

56.7.3.5.3 Scan Commands

56.7.3.5.3.1 Scan Bit (SCNB)

This is a 3-part command. The operand chooses the bit to be read or written by the CR-Scan.

Table 56-20. Scan Bit (SCNB)

SCNB			Scan Bit		
Opcode	Operand	CR-Scan	yyyyy	Description	R/W
01000	yyyyy	-	0	Scan Group Candidate (SGC)	Write only
		-	1	Conditional Group Member (CGM)	Write only
		-	2	CNFG[0]	Read only
		-	3	CNFG[1] (not implemented)	Read only
		-	4	CNFG[2] (not implemented)	Read only
		-	5	CNFG[3] (not implemented)	Read only
		-	6	Functional reset requested (FRESETR) (not implemented)	Read only
		-	7	Series topology detect	Write only
		-	15 – 8	Reserved	Read only

56.7.3.5.3.2 Scan String (SCNS)

This is a 3-part command. The operand chooses the string to be read or written by the CR-Scan.

Table 56-21. Scan String (SCNS)

SCNS			Scan String		
Opcode	Operand	CR-Scan	yyyyy	Description	R/W
01001	yyyyy	-	0	RDBACK[0] Register [31:0]	Read only
		-	1	RDBACK[1] Register [31:0] (not implemented)	Read only
		-	2	CNFG[0] Register [31:0]	Read only
		-	3	CNFG[1] Register [31:0] (not implemented)	Read only
		-	4	CNFG[2] Register [31:0] (not implemented)	Read only
		-	5	CNFG[3] Register [31:0] (not implemented)	Read only
		-	6 - 7	Reserved (read logic 0)	Read only
		-	31 - 8	Private – available for customization	

56.7.4 EPU operating states

The TAP.7 controller and STL share the use of ZBSs. The EPU has three operating states as shown in the following figure:

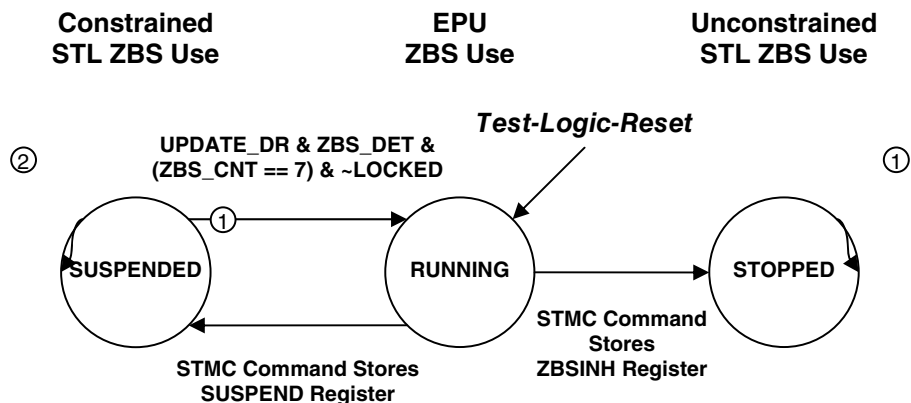


Figure 56-9. EPU operating states

The values of the SUSPEND and ZBSINH Registers determine the type of operation as shown in the following table. The Test-Logic-Reset state sets the SUSPEND and ZBSINH Register values to a logic 0.

Table 56-22. ZBS use state characteristics with both TAP.7 and other ZBS use

ZBSINH	SUSPEND	EPU operating State	ZBS use by:	Until
0	0	RUNNING	EPU	Command allocates to STL
0	1	SUSPENDED	STL	ZBS sequence allocates to EPU
1	0	STOPPED	STL	Test-Logic-Reset state allocates to EPU
1	1	Not possible		

56.7.5 System and EPU Paths

The System path is utilized when the external tool requires access to system (STL) resources. This path is used when either the SUSPEND Register is a logic 1 or the ZBS count is less than or equal to one. The EPU path is used otherwise.

The EPU scan path is constructed from five types of EPU scan paths:

- **Bypass** – A 1-bit scan path
- **Bit** – Supports set and test bit operations (a 1-bit shift register)
- **String** – Support 32-bit reads or 32-bit writes
- **Enumerate** – A 36-bit scan path that transports a TAPC address with T3 and above TAP.7s
- **Auxiliary** – An n-bit scan path that transports 1 – n bits using a conventional IEEE 1149.1 style DR-Scan path using Control Levels 4 and 5.

56.8 APU (Advanced Protocol Unit) Operation

56.8.1 Overview

The Advanced Protocol Unit (APU) is placed between the EPU and the RSU to create the T4 TAP.7 controller. The APU delivers advanced capabilities with a number of operating modes (new scan formats) that utilize the Standard Protocol and the Advanced Protocol in both the narrow and wide T4 TAP.7 versions.

The APU enables scan transfers with two signals by serializing the information related to a TAPC state machine state. It provides a number of serialization formats to balance scan flexibility and scan performance. It multiplexes the use of the TMSIC signal for T4 TAP.7 controller functions. The serialization is bypassed to provide T3 TAP.7 capability.

With a wide TAP.7, the chip architect may choose to implement the TDIC and TDOC signal functions as fixed functionality (the T3 TDIC and TDOC functions) or as programmable between the T3 TDIC and TDOC signal functions and an alternate function using a TAP.7 controller register. In this case, the default signal function may be either of the programmable choices. Programmable TDIC and TDOC functionality provides a means to fully utilize these signals in any scan topology as the T3 TAP.7 functionality is fully supported.

The entire EPU infrastructure (i.e., ZBS detection, control-level management, and command generation) is utilized when either the Standard Protocol or the Advanced Protocol is used, as these TAP.7 controller attributes are controlled entirely from the TAPC state machine state progression. The EPU is unaware of the APU's existence. All T4 features are managed outside of the EPU.

The TAP.7 serialization schemes are specified with scan formats that complement the JScan 0-3 Scan Formats inherited from the T3 TAP.7. The scan formats are called advanced scan formats as they support two-signal operation.

Two groups of scan formats (MScan and OScan0-7) provide a number of options for the serialization of scan information for use in a Star-2 scan topology. The OScan Scan Format is broken into two groups with the same function, one group has better performance but may only be used with an external tool sourced TCKC. The external tool chooses a scan format that matches the constraints imposed by the application, chip components, and possibly the external tool itself.

The MScan and OScan formats address Test and Debug use cases. The mandatory MScan and OScan0-1 formats emphasize flexibility over performance. The remaining optional Oscan2-7 formats emphasize performance over flexibility by not transmitting unneeded information such as TDI and TDO in non-shift TAPC state machine states.

The optional capabilities added with T4 and above TAP.7s change the bit sequences seen at the TAP.7 signals. A TAP.7 controller comprehends only the bit sequences generated by the use of the features it supports. It is never exposed to bit sequences it does not comprehend. A check for supporting an enabled feature is made before the feature is used (alters the bit sequences). When a T4 TAP.7 controller detects the use of an unsupported scan format, it halts its operation and places itself in a state where its operation may be resumed. This state is henceforth called "offline". A TAP.7 controller that is not offline is henceforth called "online". While offline the TAP.7 controller awaits the detection of a

special signaling sequence called the Online Activation Protocol. This protocol sequence places an offline TAP.7 controller online and synchronizes the operation of all online TAP.7s.

The Advanced Protocol Unit supports the stall of a transfer by both the external tool and STL at any TAPC state.

The APU provides the following functions:

- Bypass (for operation with the Standard Protocol)
- Scan control (for SP use)
- Configuration Control (for CP use)

The Scan Control, Configuration Control, and Transport Control functions are idled when the Standard Protocol is used as the Bypass Function connects the TCKC and TMSC pins to the EPU.

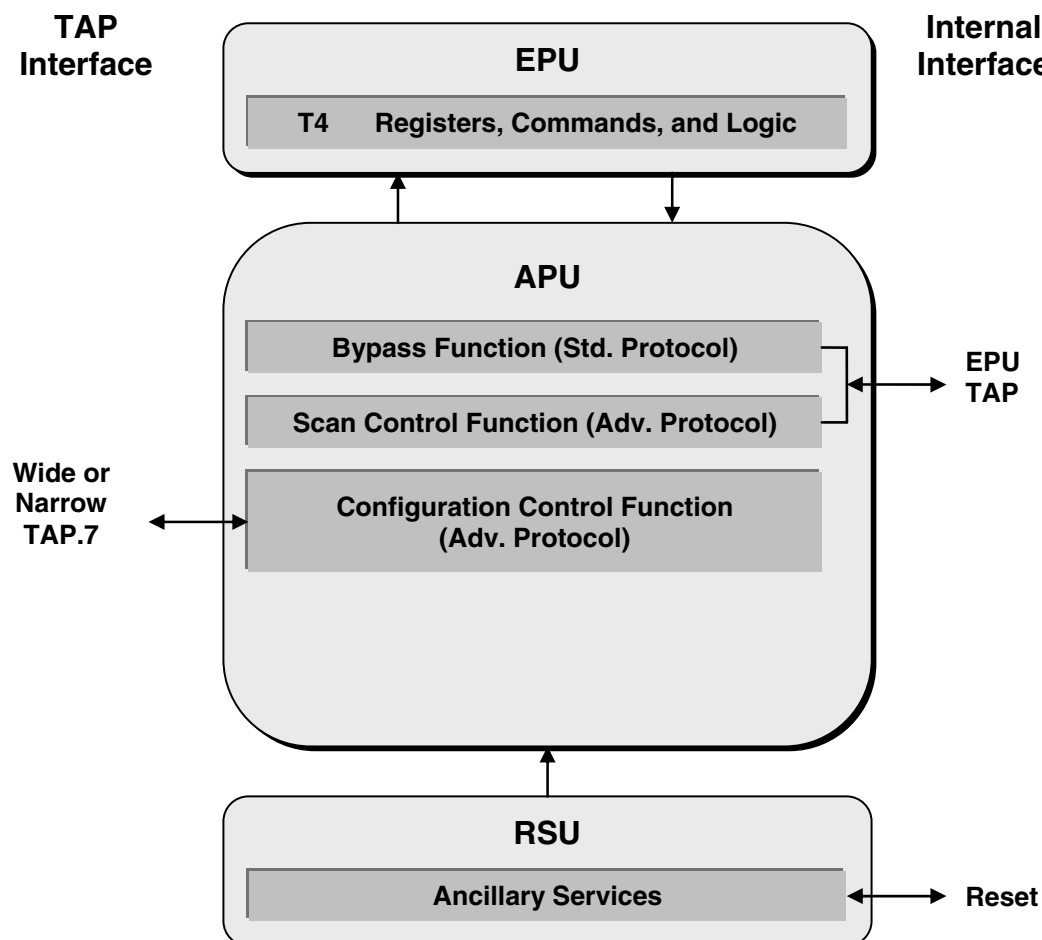


Figure 56-10. Conceptual block diagram of the APU functions

The APU's internal interface includes a connection to EPU registers, the EPU's TAP interface, connections to chip-level data channels (when they are implemented), and reset signaling with the EPU. The APU's TAP interface provides mutually exclusive use of the TMSC pin for:

- Bypass Function
- Scan Control Function
- Configuration Control Function

When the Standard Protocol is used, the APU's Bypass function connects the EPU directly to the TAP.7 pins. When the TDIC and TDOC pin functions are not available (narrow TAP.7 or TDIC and TDOC pin functions are an alternate function), the APU sources a logic 1 to the EPU inputs associated with these pins.

When the Advanced Protocol is used, the APU multiplexes the use of the TMSC pin between the other APU functions. Scan Control acts as a serial to parallel converter, translating the parallel information at the EPU's TAP interface into bidirectional serial transfers. Since the Scan Control is literally a protocol adaptor, the entire EPU infrastructure (ZBS detection, control-level management, and command generation) is utilized when either the Standard or Advanced Protocol is used, as these TAP.7 controller attributes are controlled entirely from the TAPC state progression.

56.8.2 Operation

The conceptual view of the APU operation is shown in the following figure.

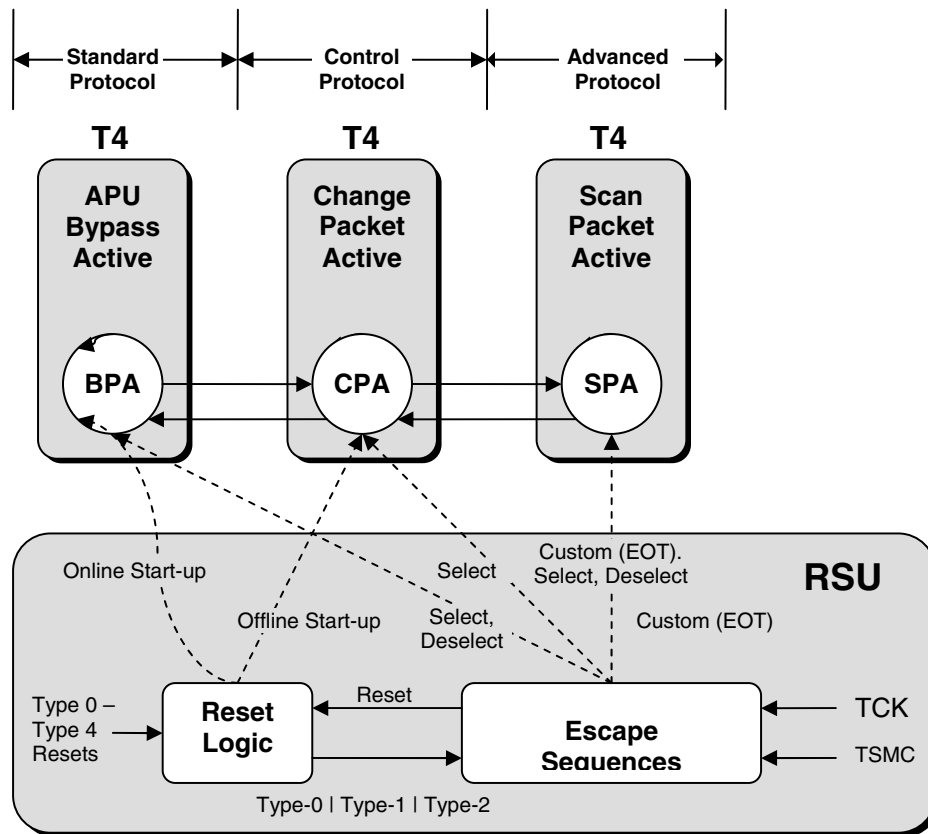


Figure 56-11. Conceptual view of APU operation

The APU operating states managing the use of the TAP pins and the ancillary (RSU) services use of the TMS pin.

APU operating states:

- **BPA** – ByPass Active
- **CPA** – Change Packet Active
- **SPA** – Scan Packet Active

56.8.2.1 BPA

The *BPA* state allocates the use of the TMS pin to the Standard Protocol. This state is entered two ways:

- As a result of a TAP.7 controller reset when the TAP.7 controller starts up online.
- From the CPA state when a TAP.7 controller register write specifies the use of the Standard Protocol.

56.8.2.2 CPA

The *CPA* state allocates the use of the TMSC pin to a CP. This state is entered in one of three ways:

- As a result of a TAP.7 controller reset when the TAP.7 controller starts up offline.
- A value specifying the use of the Advanced Protocol is written to the SCNFMT Register when using the Standard Protocol.
- A TAP.7 controller register write occurs when using the Advanced Protocol.

56.8.2.3 SPA

The *SPA* state allocates the use of the TMSC pin to an SP. This state is entered when the SCNFMT Register specifies the use of the Advance Protocol when a CP ends.

56.8.3 Escape sequences

Escape-sequence detection provides services to the Reset Logic, Scan, Configuration Control, and Transport Control functions as outlined below:

- **Scan and Transport Control** – End of Transfer (EOT) escape sequence detected.
- **Configuration Control** – Synchronize Advanced Protocol (SAP) escape sequence.
- **Reset Logic** – Reset (RES) escape sequence detected.

An EOT escape sequence is used to increase the efficiency of OScan Scan formats.

An SAP escape sequence is used to place TAP.7 controllers online.

A RES escape sequence asynchronously causes generation of a Type-3 Reset. Following a RES escape sequence, the drive of the TMSC pin is inhibited so it is not driven when TCKC becomes a logic 0 immediately following the RES escape sequence.

56.8.4 Signal behaviors

The TCKC signal:

- The test clock

The TMS signal:

- An input when the Standard Protocol is used
- Bi-directional when the Advanced Protocol is used

The TDI and TDO signals:

- Deleted with a narrow TAP.7
- One of two functions with a wide TAP.7
 - Fixed – as the TAP.7 TDIC and TDOC pin functions, or
 - Programmable – as the TAP.7 TDIC and TDOC signal functions or alternate functions with a TAP.7 controller reset establishing the default signal function.

56.8.5 APU Functions

56.8.5.1 Bypass Active (BPA) Function

When the Standard Protocol is used, the APU connects the EPU inputs and outputs directly to the TAP.7 pins. If the TDIC and TDOC pin functions are not present (e.g., a narrow TAP.7 or the TDIC and TDOC pins have an alternate function assigned to them), the APU provides a logic 1 to the EPU for unsupported pin functions and ignores the EPU's TDO output data.

56.8.5.2 Scan Packet Active (SPA) Function

The Scan Control function is encapsulated within the Scan Packet Active (SPA) state. With the Standard Protocol, the external tool and TAP.7 exchange the scan information associated with a single TAPC state machine state in one TCK period using the TMS(C), TDI(C), and TDO(C) pins. With the Advanced Protocol all or part of this information is exchanged serially within a Scan Packet (SP) as shown in . There is a one to one correspondence between SPs and TAPC state machine state changes, except when the SPA state precedes the CPA state. In this case the SP preceding a CP does not advance the TAPC state machine state.

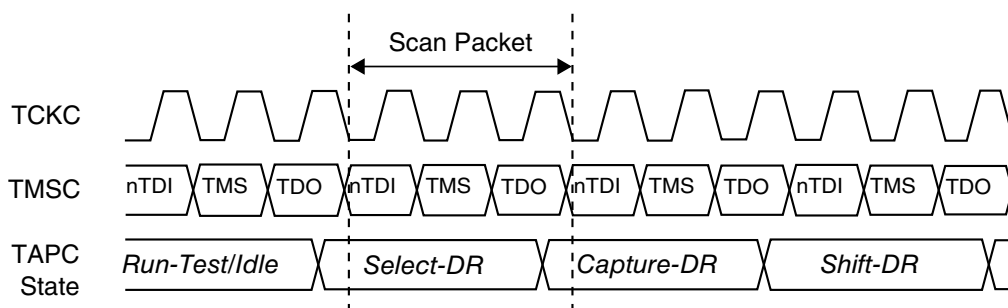


Figure 56-12. Scan Packet Sequence

A comparison of the parallel transfer of TAP information with the Standard Protocol and the serialization and de-serialization of the TAP information using the TCKC and TMSC pins with the Advanced Protocol is shown in FIGURE. The IEEE 1149.1 signals between the EPU and STL are prefixed with SYS_.

The APU manages the serialization and de-serialization of the scan information. The APU converts the TDI and TMS information within a Scan Packet (SP) into the TMS and TDI information presented to the EPU. It converts the combination of the EPU TDO and EPU TDO_OE signals into the TDO information in the same SP. If the EPU indicates that the TDO pin drive is not enabled, logic 1 TDO data is created, otherwise the TDO data is the supplied by either the EPU or STL.

Information within a Scan Packet is first passed from the external tool to the TAP.7 followed by information passed from the TAP.7 to the external tool. In this example the scan format specifies the exchange of the TMS, TDI, and TDO TAPC information. Each TAPC state machine state takes three TCKC periods. Other scan formats specify the exchange of different mixes of information. Control information is added with the most flexible scan formats. Some scan formats send less information and consequently take fewer TCKC periods.

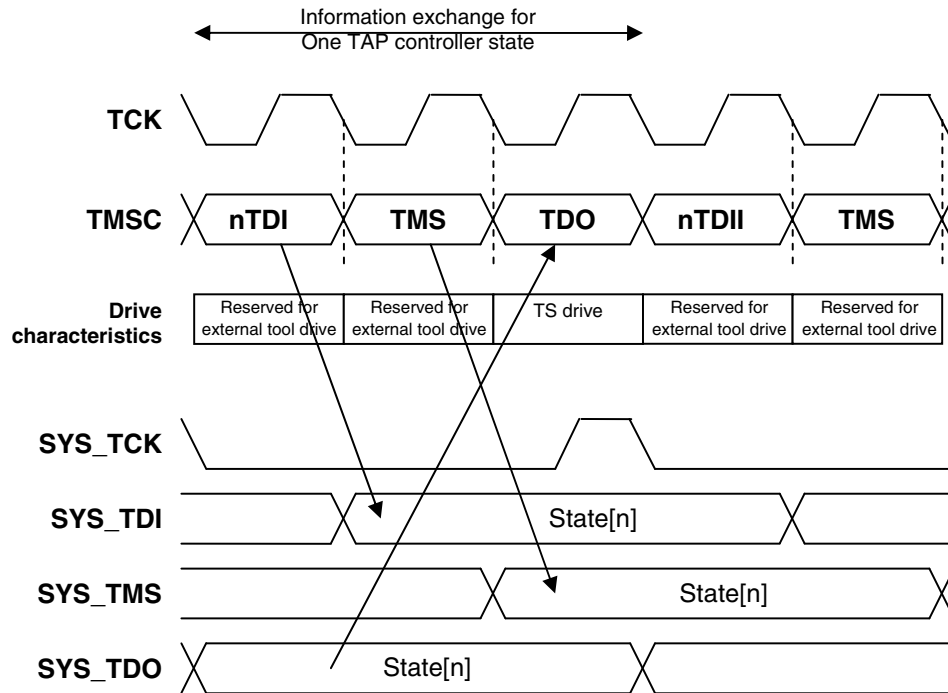


Figure 56-13. Serialization and De-serialization of Scan Packet Information

In order to produce comparable TAPC state machine state rates with a TAP.1 and TAP.7 controller:

- Full-cycle timing between the external tool and TAP.7 allows operation at higher TCKC frequencies.
- The number of bits transferred to advance a TAPC state machine is minimized. (reaching as few as one for some scan formats.)

The TAP.7 controller provides both full-cycle bit timing (falling TCKC edge to falling TCKC edge) and half-cycle bit timing (falling TCKC edge to rising TCKC edge), with a TAP.7 register used to define the type of timing used. Half-cycle timing is used as the default following a TAP.7 controller reset. The TCKC frequency is reduced to accommodate half-cycle timing.

With full-cycle timing, the TMSC input and output data are both output with and sampled with the falling edge of the TCKC. This means that the entire TCKC period may be spent transferring a bit from the external tool to the TAP.7 or vice-versa. The TCKC frequency can therefore be maximized when full-cycle timing is used. With half-cycle timing, the APU samples the TMSC input data with the rising edge of the TCKC and outputs TMSC data with the falling edge of TCKC.

Minimizing the number of bits transferred per TAPC state machine state is affected by a number of factors. The information that must be transferred per TAPC state machine state varies for different test and debug applications. Also, with some test and debug

applications, all TAPC state machine states do not require that the same amount of information be transferred. This is especially true with debug applications where specialized use of the TAP is more likely.

56.8.5.3 Control Process Active (CPA) Function

While using the Advanced Protocol, all TAP.7 configuration changes occur within the CPA state. All online and offline transitions also occur while in the CPA state. When entering the CPA state from the BPA or SPA state, the CPA state checks for the use of supported features affecting the Advanced Protocol. The CPA state is exited when the enabled features affecting the Advanced Protocol are supported.

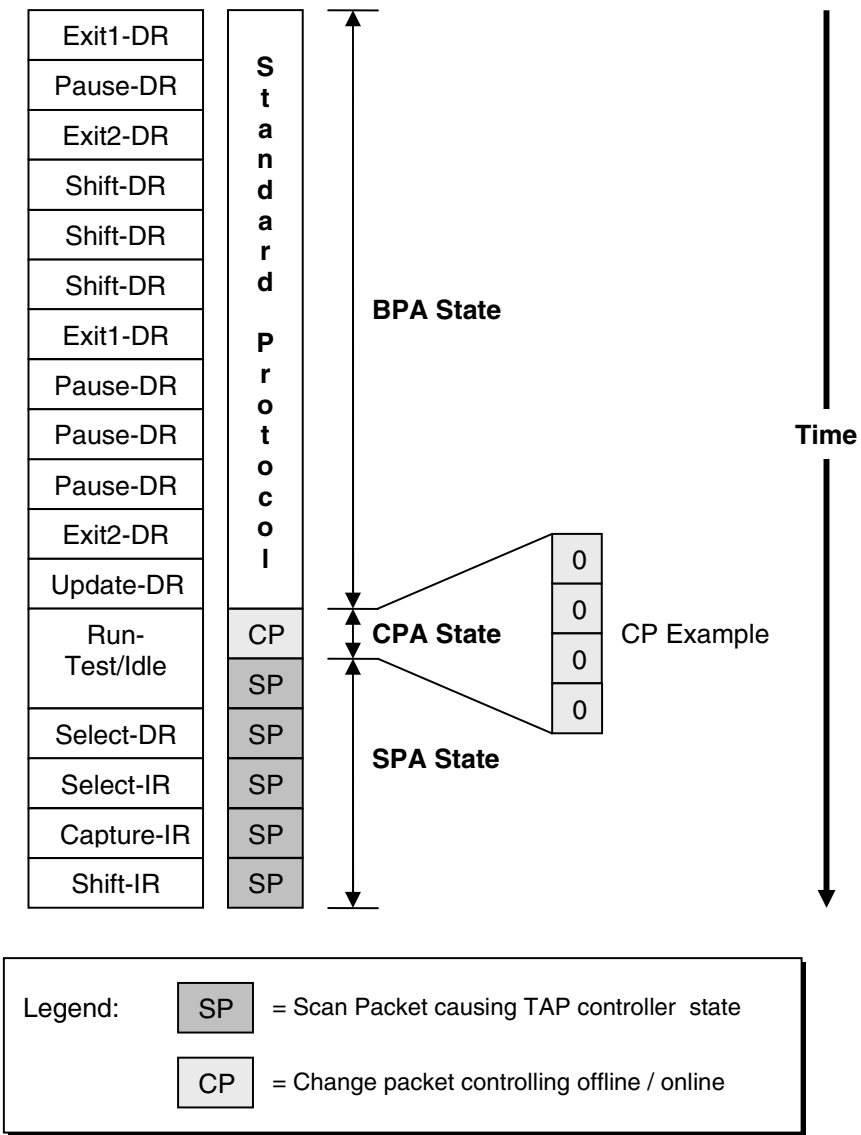


Figure 56-14. CPA State Entry from BPA State with Continued Online Operation

An SP preceding a CP does not advance the TAPC state machine state. This SP performs housekeeping functions that assure the contents of the SP preceding this SP have been fully utilized. This has significance when the SP contains control information that allows the STL to stall the completion of a SP.

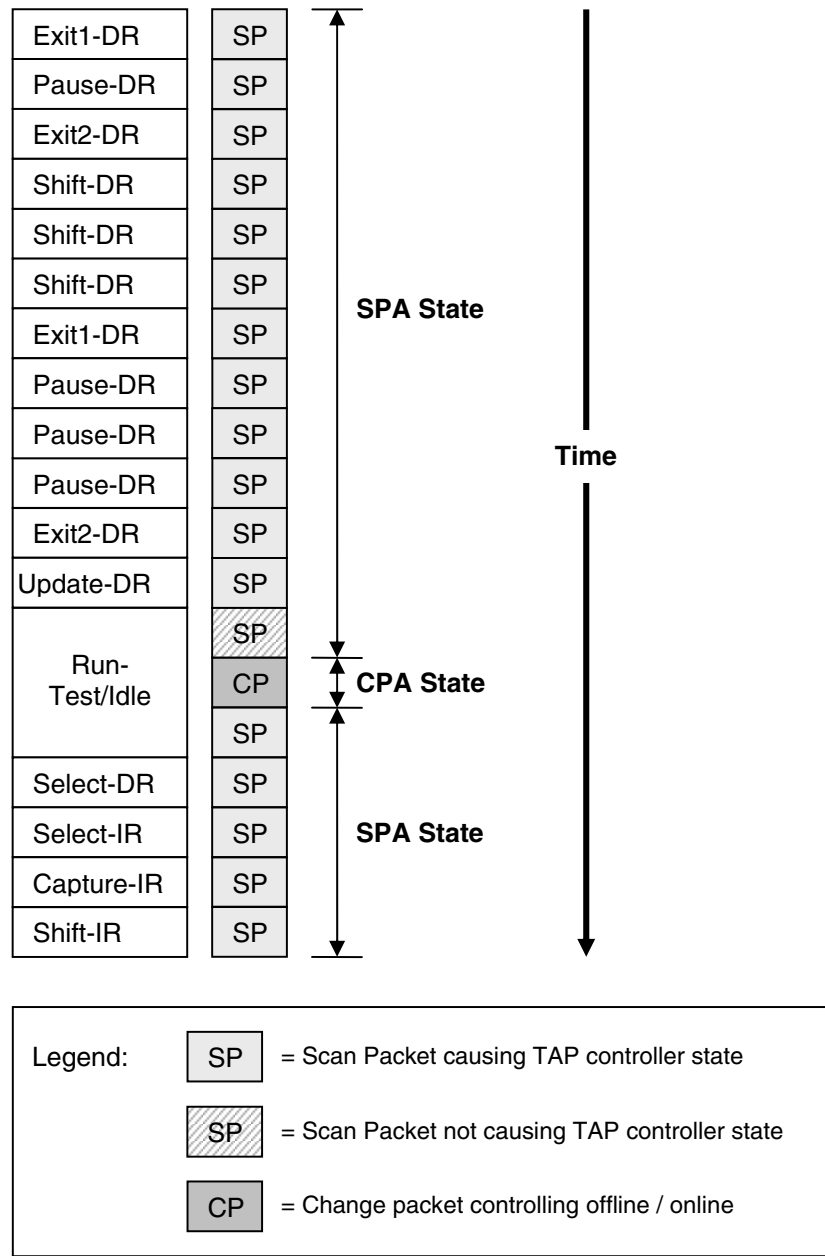


Figure 56-15. CPA State Entry from the SPA State with Continued Online Operation

When an unsupported feature is enabled, the TAP.7 controller is placed offline. This suspends the TAP.7 controller's operation at a point where its operation may later be synchronized to the remainder of the system. The APU stops advancing the EPU's and CLTAPC's TAPC state machine state while offline. It also suspends Data Channels activity while offline. The detection of the appropriate Online Activation Protocol while

the TAP.7 controller is offline places the TAP.7 controller online and causes an exit from the CPA state to the SPA state. An offline to online transition initiates the use of a predefined set of mandatory features in all online TAP.7 controllers to assure they operate synchronously.

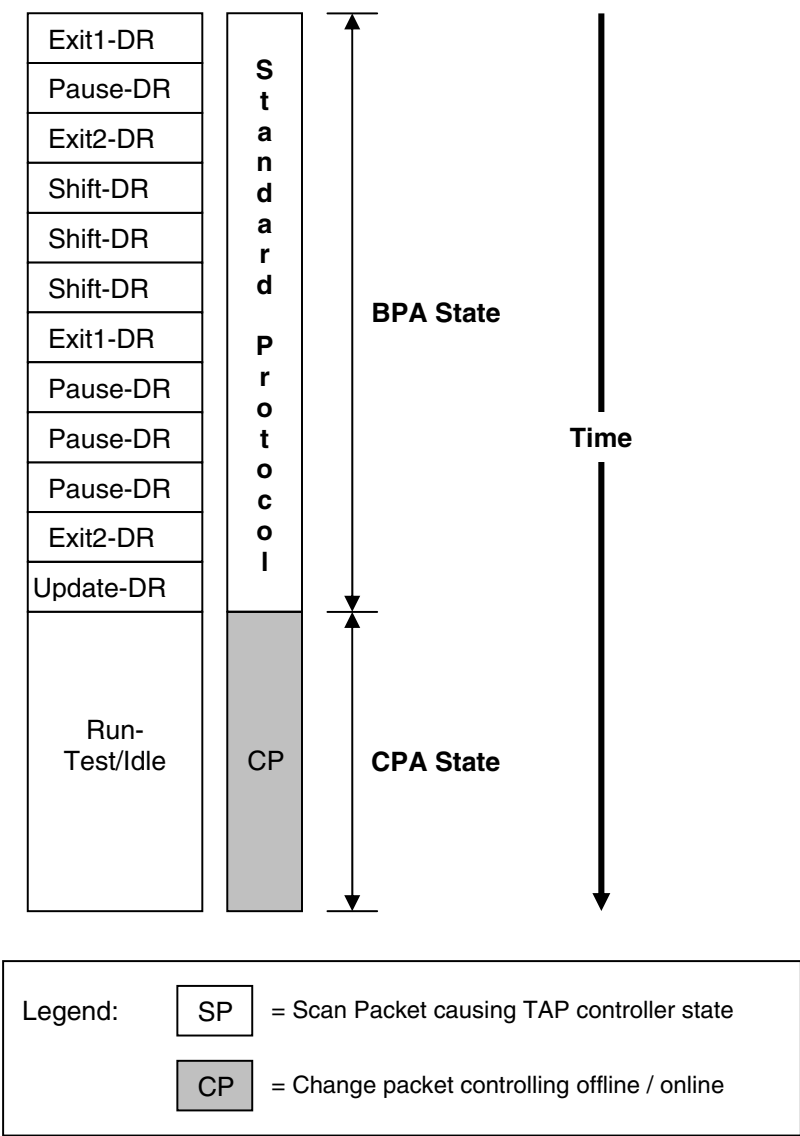


Figure 56-16. CPA State Entry from the BPA State with Offline Operation

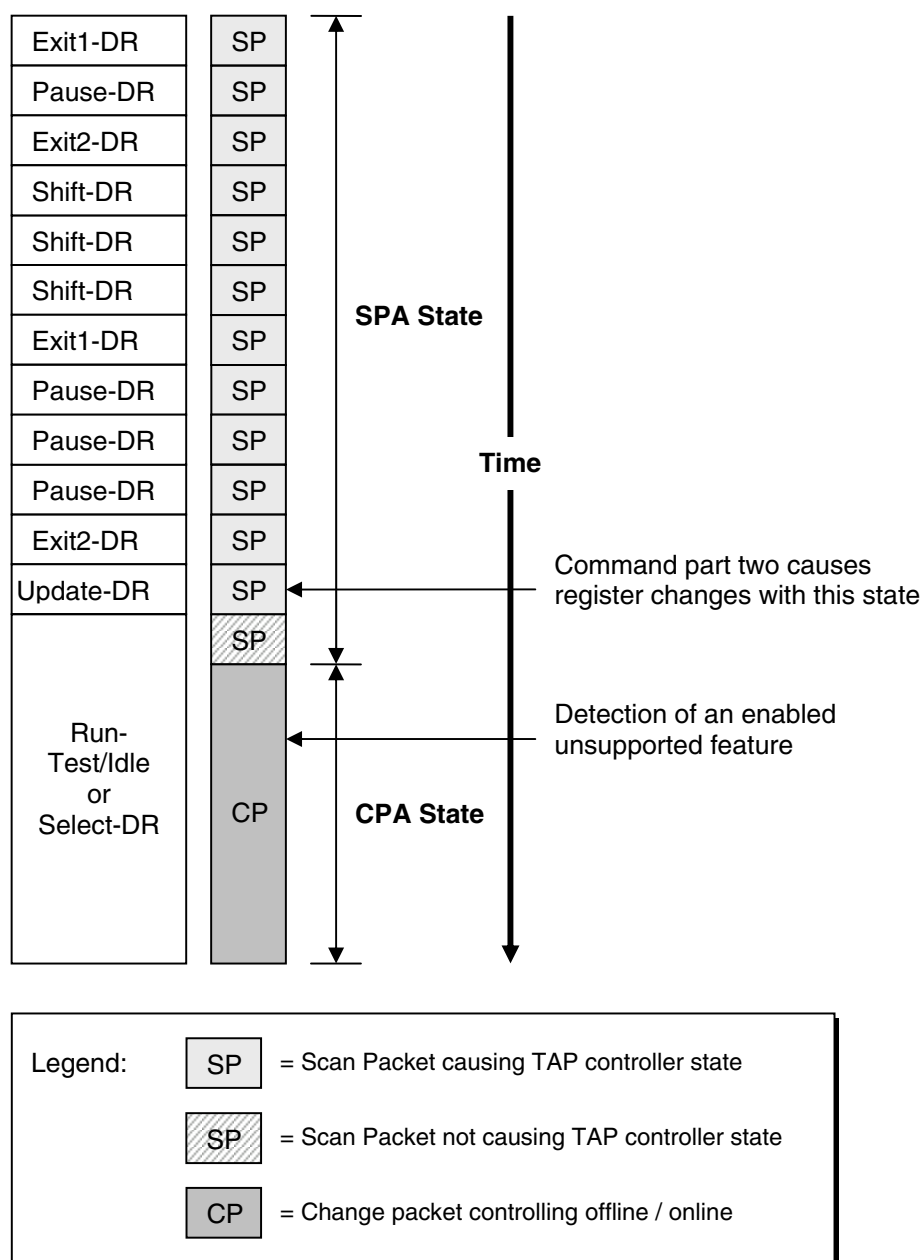


Figure 56-17. CPA State Entry from the SPA State with Offline Operation

A TAP.7 controller reset places the TAP.7 controller offline when the offline at start-up option is used. From this point the TAP.7 controller awaits being placed online.

56.8.6 Configuration Change Packets (CP)

A Change Packet (CP) is composed of three element types:

- **Preamble** – A 1-bit value that is a don't care. This bit facilitates a Standard to Advanced Protocol change with some external tool implementations.

- **Body** – Two or more bits representing one or more CP directives
 - **00** – CP_END
 - **01** – CP_NOP (extends the body by 1-bit)
 - **10** – CP_NOP (extends the body by 1-bit)
 - **11** – CP_RES (Generate a Type-3 TAP.7 controller reset coincident with the postamble bit.
- **Postamble** – A 1-bit value that is a don't care. This bit facilitates and Advanced to Standard Protocol change with some external tool implementations.

A minimum-length CP has a Preamble, a 2-bit Body, and a Postamble. This can be as simple as four logic zero values.

56.8.7 Scan Packet

A Scan Packet defines the period of TMS pin activity where control information exchanged by the external tool and TAP.7 controller affects the TAPC state, except when the SP precedes a Change Packet. The external tool should view the Scan Packet as:

- Exchanging only the necessary TAPC information (TMS, TDI, and TDO or some subset thereof)
- Causing an advance of the TAP.7 TAPC state when it is not followed by a Change Packet.
- Causing an advance of the STL TAPC state when it is not followed by a Change Packet and when the STL is coupled.

With this perspective, the external tool views Scan Packets as running a virtual TAPC state machine at the TAP.7 pins. The effects of each Scan Packet take effect when the Scan Packet completes. The real TAP.7 and STL controller state machine states may change state before or after the virtual state machine.

56.8.8 SP Format

A SP serially exchanges the TDI, TMS, and TDO or a subset thereof, between the external tool and TAP.7 controller. Control information may be added to this exchange. A SP has three parts:

- **Payload** – Data + Control information (all scan formats), the payload content is varied to trade-off performance and flexibility. RDY bit(s) within the payload are included with some scan formats to provide the STL a means to stall progression of the TAPC state.
- **Delay** – Control information (optional) providing separation of adjacent payloads in time. Delay elements support simple external tool implementations where a delay element is used to stall progression of the TAPC state while it develops a new SP following the completion of the prior payload.

The header elements, delay elements, and control information within the payload element is consumed by the TAP.7 controller. This and other control information within the SP payload is not visible to the EPU and STL TAPCs.

56.8.8.1 Payload Element Content

56.8.8.1.1 Input and Output Bit Frames

An input bit frame is sourced by the external tool and an output bit frame is sourced by the TAP.7 controller. The payload may contain an input bit frame, an output bit frame, or both. When an input bit frame arrives at the TAP.7 controller, the TDI and TMS information within it is presented in parallel to the TAP.1 controllers within the EPU and STL. The output bit frame begins when the input frame is completed. The payload is completed when the EPU and STL are ready to utilize this information and the TDO data needed to complete the output bit frame is available. This process is repeated for each SP.

With input bit frames, MScan and OScan scan formats have no control bits input bit frames.

With output bit frames:

- TDO information passed to the external tool is a logic 1 when the EPU indicates the TDO pin would be high impedance.
- RDY bits provide TAP.7 stall opportunities with MScan and some OScan formats.
- PC0 and PC1 bits are added to MScan scan formats to allow Wire-ANDed output.

56.8.8.1.2 Varying the content of input and output bit frames

The various scan formats (MScan, OScan0-7) provide a number of options for the serialization of scan information for use in a Star-2 scan topology. These scan formats vary the content of SPs to balance flexibility and performance for different use cases. The amount of information transferred per TAPC state is based on the constraints imposed by these use cases. The MScan and OScan scan formats determine the factors affecting the payload content as shown in the following table.

Table 56-23. Factors determining the SP payload content

Scan format	Payload content determined by the:	
	TAPC state	Header
MScan	–	–
OScan	Yes	–

56.8.8.2 Delay Element Content

A delay element can be zero, one, two, or n bits in length. One and two bit delay elements are fixed length with neither the external tool or TAP.7 controller driving the TMSC pin. The TMSC pin is kept at the last value driven by either the external tool or TS. With an n-bit delay elements, the external tool drives the TMSC pin for the first n-1 bits and may drive the TMSC pin the nth bit. With an n-bit delay element the string of delay element bits is a series of directives virtually the same as those used by a CP (DLY_NOP, DLY_END, and DLY_RES).

56.8.8.3 Scan Formats

56.8.8.3.1 Overview

Scan formats utilizing the Advanced Protocol address different use cases. The amount of information transferred per TAPC state is based on the constraints imposed by these use cases. In one extreme two types of control information, the first allowing the stall of the completion of the Scan Packet, and the second supporting the Wire-AND of TAP.7 controller output are added to the TMS, TDI, and TDO information within the Scan Packet. In the other extreme, the external tool may minimize the information transferred for each TAPC state using its understanding of the application and transfer only the data of interest. In some cases the transfer of only one bit of information per TAPC is

required. There are a number of scan formats that operate between these two extremes. The external tool chooses a scan format that best matches the performance/functionality needs of the application.

The scan formats added to specify the use of the Advance Protocol are listed below:

- **MScan** – Maximum flexibility.
- **OScan0-7** – Optimized for test and debug applications.

These scan formats provide seven different functions. Five factors have influenced the definition of these scan formats:

- Supporting the capabilities of the IEEE1149.x and IEEE 1532 Standards.
- Providing the STL and external tool an opportunity to stall the TAPC state progression.
- Providing a minimum pin count and scalable functionality.
- Maximizing the efficiency of scan operations utilized for applications debug operations.
- Providing a rich set of capability for high end applications debug.

The performance and flexibility of MScan and OScan scan formats are shown in [Table 56-24](#) and [Table 56-26](#). The performance and flexibility of these scan formats are given subjective/approximate ratings in these tables. These ratings range from least to best. Use cases for these scan formats are also included in these tables.

56.8.8.3.2 MScan scan formats

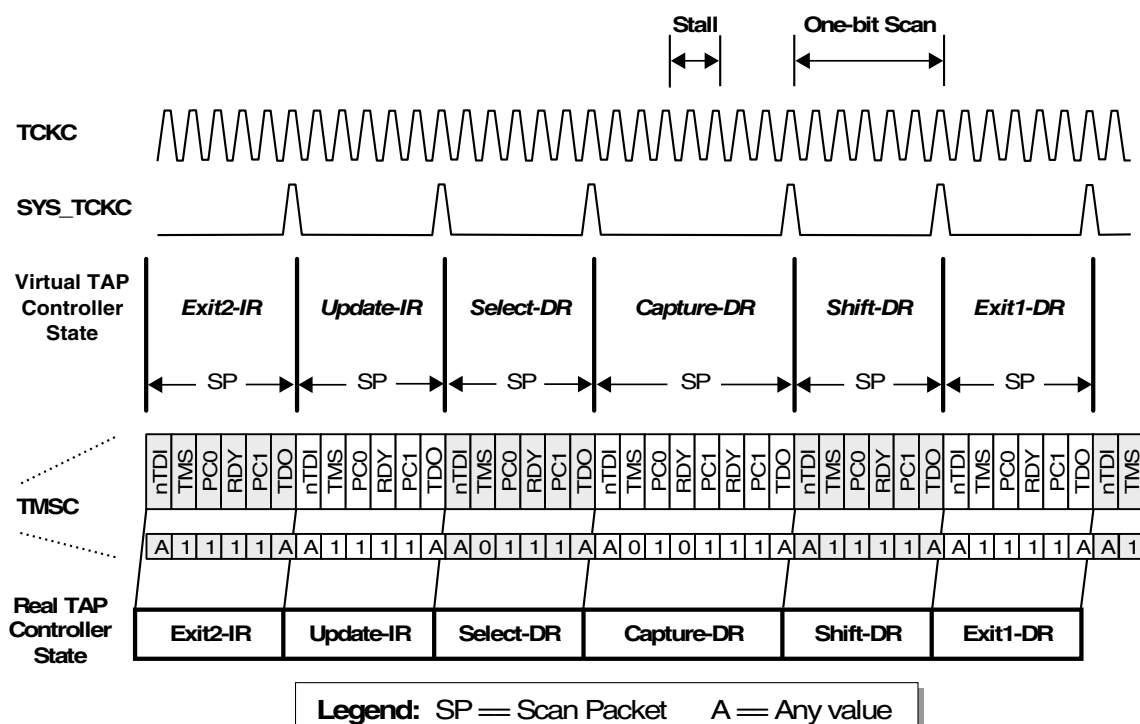
The MScan scan format provides a universal scan function. It is the most flexible but lowest-performance scan format. The Scan Packets used by these scan formats are packed with data and control information and is the same for all TAPC states. This scan format accommodates the deployment of IEEE 1149.1 IP with non-compliant behavior within the STL. The STL is allowed to stall the TAPC state progression (to pace the arrival of Scan Packets with its ability to process them), and uses Wire-ANDed Output.

Table 56-24. MScan scan format use case summary

Scan format	Supporting	Performance	Flexibility	TCKC source
MScan	STL Stalls	Least	Best	External tool or TS
	Test			
	Non-compliant behavior STL IP			
	Multi-chip debug communication			
	Controller ID Allocation			

Table 56-25. MScan Scan Packet content

	Non-shift TAP controller states							Shift TAP controller states					
MScan	nTDI	TMS	PC0	RDY	PC1	TDO		nTDI	TMS	PC0	RDY	PC1	TDO

**Figure 56-18. MScan Transaction**

56.8.8.3.3 OScan Scan Formats

The OScan scan formats provide optimization choices where no knowledge of the data content of the scan exchange is needed. The Scan Packet content of these scan formats is modulated by a combination of the scan format and the TAPC state for some OScan scan formats. For instance, TDI data and TDO data is not exchanged for non-shift TAPC states and included in shift TAPC states in certain OScan formats. The modulation of Scan Packet content with TAPC state varies by scan format.

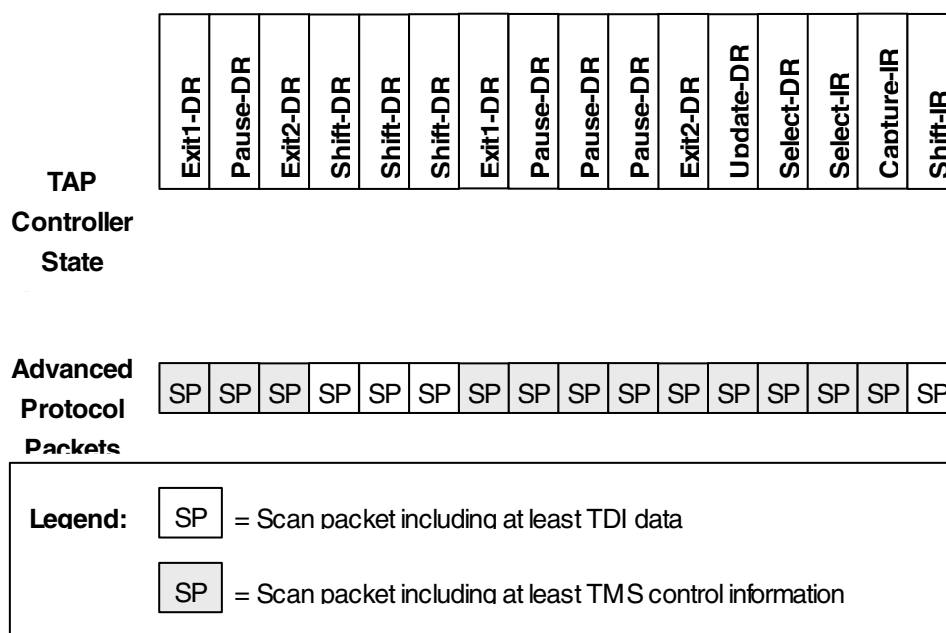


Figure 56-19. OScan Scan Packet content/TAPC state relationships

OScan scan formats replicate four functions in two forms:

- **OScan0 – OScan3:** Providing the best performance for a test clock source by the target system.
- **OScan 4 – OScan7:** Providing the best performance for a test clock sourced by the external tool.

The OScan4 – OScan7 scan formats are more efficient, as EOT escape sequences are used to identify an exit from the Shift-xR TAPC state. A Scan Packet associated with the Shift-xR TAPC state does not include a TMS bit. Instead, and EOT escape sequence overlaid on an nTDI bit causes an exit from shift TAPC state with no exit from this state occurring otherwise. A look at the 32-bit DR-Scan shows the impact of this optimization. A total of 96 bits are required when each Scan Packet associated with a Shift TAPC state when TMS is included in these Scan Packets while a total of 64 bits are required when TMS is not included in these Scan Packets. Assuming the EOT escape adds the equivalent of three bit periods, the number of TCK periods for this transfer is 67/96 or roughly 70% of the total without this optimization.

These scan formats may only be used with an external tool sourced TCKC as EOTs are utilized to improve its performance. The OScan0 – OScan3 Scan Formats always have a TMS bit included in the Scan Packet. The OScan scan format use case summary is shown in the following table.

Table 56-26. OScan scan format use case summary

Scan Format	Supporting	Performance	Flexibility	TCKC Source
OScan0	Stalls	↓	Best	External tool
OScan1	Test		↑	
OScan2	Debug Performance			
OScan3	Debug Downloads (input only)	Best	↑	External tool or TS
OScan4	Stalls	↓		
OScan5	Test			
OScan6	Debug Performance			
OScan7	Debug Downloads (input only)	Best		

Certain OScan optimizations are applied in a hierarchical manner.

- **Optimization I:** OScan1/OScan5 – RDY bit(s) and their trailing ONE bits are deleted from all Scan Packets.
- **Optimization II:** OScan2/OScan6 – Optimization I + nTDI and TDO bits are deleted from Scan Packets associated with non-shift TAPC states.
- **Optimization III:** OScan3/OScan7 – Optimization II + TDO bits are deleted from Scan Packets.
- **Optimization IV:** OScan4 – OScan7 – TMS bits are deleted from Scan Packets associated with shift TAPC states.

Table 56-27. OScan Scan Format "Payload"

Scan Format	Non-shift TAP controller states					Shift TAP controller states			
OScan0	nTDI	TMS	RDY	TDO		nTDI	TMS	RDY	TDO
OScan1	nTDI	TMS	–	TDO		nTDI	TMS	–	TDO
OScan2	–	TMS	–	–		nTDI	TMS	–	TDO
OScan3	–	TMS	–	–		nTDI	TMS	–	–
OScan4	nTDI	TMS	RDY	TDO		nTDI	–	RDY	TDO
OScan5	nTDI	TMS	–	TDO		nTDI	–	–	TDO
OScan6	–	TMS	–	–	Note 1	nTDI	TMS	–	TDO
					Note 2	nTDI	–	–	TDO
OScan7	–	TMS	–	–		nTDI	–	–	–

Note 1: Following Capture-xR or Exit2-xR Scan Packet

Note 2: Following Shift-xR Scan Packet

56.9 Functional Description

This section combines the information given in the Ancillary Services, EPU Operation, and APU Operation sections into a single reference for performing common TAP.7 tasks.

56.9.1 Switching from Standard Protocol to Advanced Protocol

Upon exit of Type-0 to Type-4 reset, the CJTAG begins operation in Standard Protocol 4-pin mode. The following steps are required to move from Standard Protocol 4-pin mode to Advanced Protocol (either 4-pin or 2-pin) mode.

The first step required to move to the Advanced Protocol is to change the CJTAG control level to control level 2 via zero-bit scans (ZBS), as described in [Control levels](#).

After the control level is set to control level 2, TAP.7 commands can be executed as described in [EPU Commands](#). Execution of the Store Scan Format (STFMT) to set the scan format to any OSCAN0-7 or MSCAN format begins the switch from the Standard Protocol to the Advanced Protocol.

Once the STFMT command is executed selecting an Advanced Protocol scan format, the CJTAG transitions to the intermediate Control Protocol mode. The Control Protocol uses TMS inputs to control Configuration Change Packets as described in [Control Process Active \(CPA\) Function](#) and [Configuration Change Packets \(CP\)](#). Loading a change packet body value of 00 (CP_END) results in the switch to the Advanced Protocol.

If Advanced Protocol 2-pin operation is desired, the APFC register should be written to a value of 10 or 11 prior to the execution of the STFMT command that selects an advanced scan format. The APFC register is written via the Store Conditional 2-bit (STC2) command described in [Store Conditional 2-bit \(STC2\)](#).

Chapter 57

JTAG Data Communication (JDC)

57.1 Introduction

The JTAG Data Communication (JDC) module provides the capability to move register data between the IPS and JTAG domains. This facilitates communication between internal resources that access memory mapped register space and an external tool that accesses the JTAG port.

57.2 Overview

The JDC module consists of IPS accessible registers, JTAG accessible registers, and associated logic to coordinate movement of data from one register domain to another.

The JDC implements the following IPS data registers, occupying separate memory space:

- 32-bit memory mapped register that can be read or written via IPS (JOUT_IPS), and whose contents are ported out for capture into a JTAG register (JOUT) to be read via the JTAG port.
- 32-bit memory mapped register that can only be read via IPS (JIN_IPS), and whose contents are loaded from a JTAG register (JIN).

In addition to the data registers themselves, logic is implemented to indicate when new data has been shifted in via the JTAG port and is ready to be read from the JIN_IPS register, and when new data has been written to the JOUT_IPS register and is ready to be read via the JTAG port. The state of these flags is captured into a status register (MSR), and also provided to a JTAG register (JOUT) for tool visibility.

There is a single bus interface to port register data out to the JTAGC, and a single bus interface to port data in from the JTAGC. The architecture is shown in the following figure.

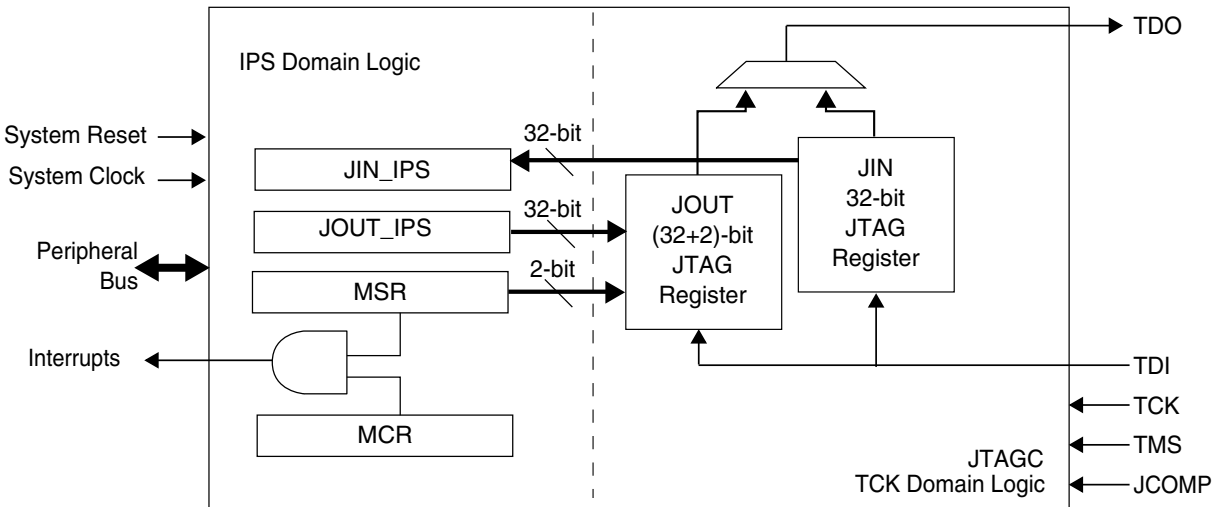


Figure 57-1. JDC block diagram

57.3 Memory mapped registers

Four 32-bit registers are implemented. Only 32-bit accesses are valid. The effects of access that are not 32 bits are not defined.

JDC memory map

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
0	Module Configuration Register (JDC_MCR)	32	R/W	0000_0000h	57.3.1/2899
4	Module Status Register (JDC_MSR)	32	R/W	0000_0000h	57.3.2/2900
8	JTAG Output Data Register (JDC_JOUT_IPS)	32	R/W	0000_0000h	57.3.3/2901
C	JTAG Input Data Register (JDC_JIN_IPS)	32	R	0000_0000h	57.3.4/2902

57.3.1 Module Configuration Register (JDC_MCR)

The MCR is used to enable the interrupt outputs of the JDC. This register is reset by system functional reset.

Address: 0h base + 0h offset = 0h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	Reserved															JIN_IEN
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	Reserved															JOUT_IEN
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

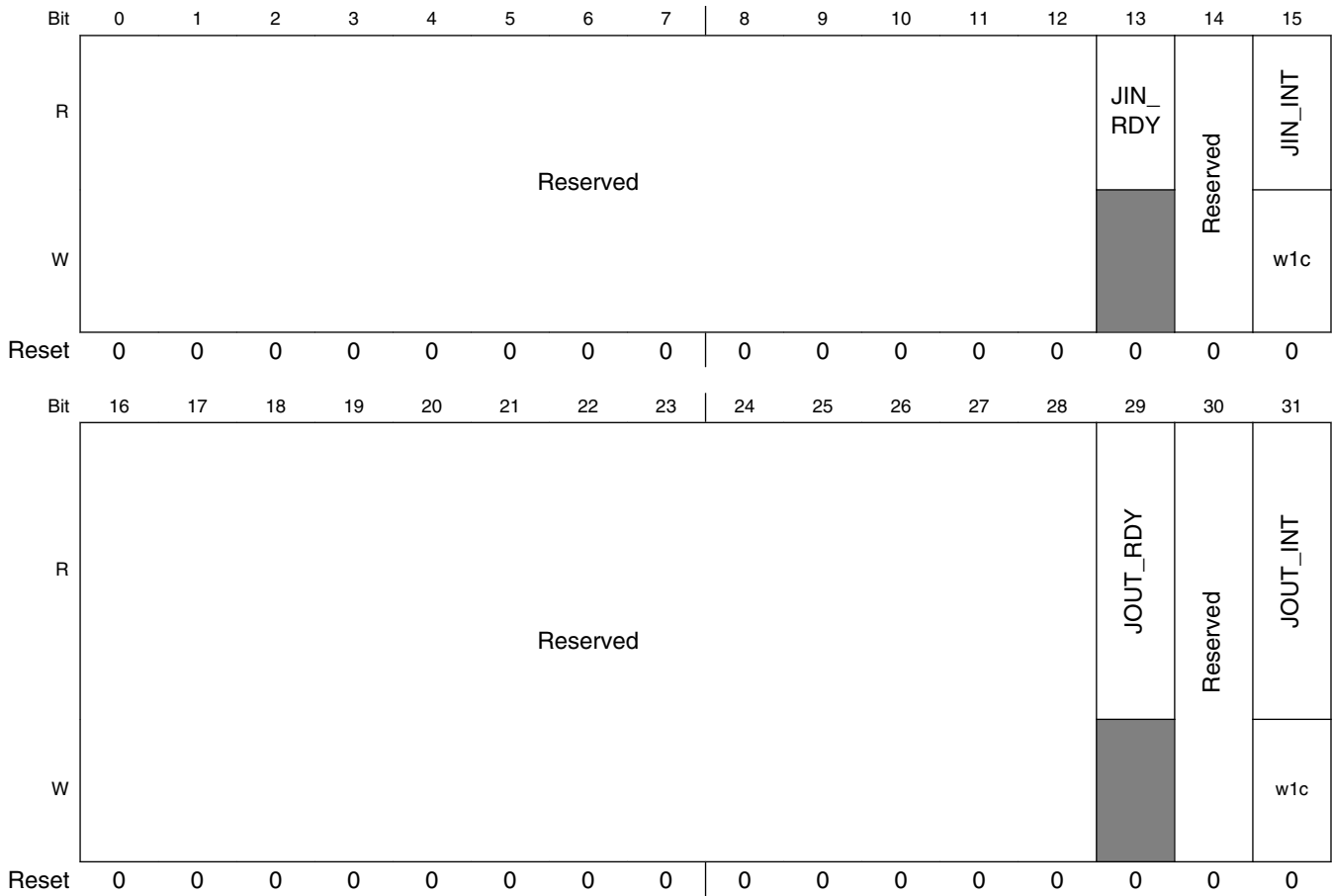
JDC_MCR field descriptions

Field	Description
0–14 Reserved	This field is reserved.
15 JIN_IEN	JIN Interrupt Enable. 0 Setting of MSR[JIN_INT] bit does not assert the JIN interrupt 1 Setting of MSR[JIN_INT] bit asserts the JIN interrupt
16–30 Reserved	This field is reserved.
31 JOUT_IEN	JOUT Interrupt Enable. 0 Setting of MSR[JOUT_INT] bit does not assert the JOUT interrupt 1 Setting of MSR[JOUT_INT] bit asserts the JOUT interrupt

57.3.2 Module Status Register (JDC_MSR)

The MSR holds the JTAG register status and interrupt bits. This register is reset by system functional reset.

Address: 0h base + 4h offset = 4h



JDC_MSR field descriptions

Field	Description
0–12 Reserved	This field is reserved.
13 JIN_RDY	JIN Ready (read only). 0 Cleared upon software read of JIN_IPS contents via IPS 1 Set when new data is written to the JIN_IPS register
14 Reserved	This field is reserved.
15 JIN_INT	JIN Interrupt.

Table continues on the next page...

JDC_MSR field descriptions (continued)

Field	Description
	0 Cleared by writing logic 1 1 Set when new data is written to the JIN_IPS register
16–28 Reserved	This field is reserved.
29 JOUT_RDY	JOUT Ready (read only). 0 Cleared upon tool read of JOUT register via JTAG port 1 Set when new data is written to the JOUT_IPS register
30 Reserved	This field is reserved.
31 JOUT_INT	JOUT Interrupt. 0 Cleared by writing logic 1 1 Set when JOUT_RDY bit is cleared by tool reading JOUT register

57.3.3 JTAG Output Data Register (JDC_JOUT_IPS)

The JOUT_IPS register holds data written via IPS. The JOUT_IPS contents are ported out and captured into the JOUT register to be read via the JTAG port. This register is reset by system functional reset.

Address: 0h base + 8h offset = 8h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R																																
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

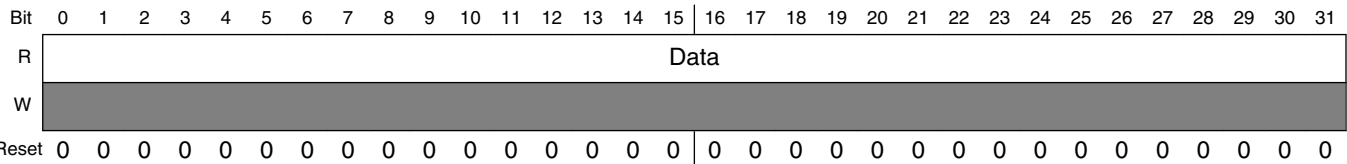
JDC_JOUT_IPS field descriptions

Field	Description
0–31 Data	JOUT_IPS data.

57.3.4 JTAG Input Data Register (JDC_JIN_IPS)

Data written to the JTAG input data register (JIN) via the JTAG port is held in the JIN_IPS register, where it can be read via IPS. This register is reset by system functional reset.

Address: 0h base + Ch offset = Ch



JDC_JIN_IPS field descriptions

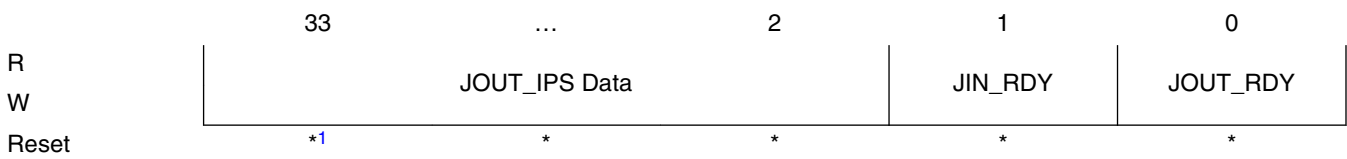
Field	Description
0–31 Data	JIN_IPS data.

57.4 Non-Memory mapped register definition

The JDC also implements two JTAG accessible registers that are not memory mapped. One JTAG register is used to shift in data to be placed in the JIN_IPS register. The other JTAG register captures data from the JOUT_IPS register and ready status from the MSR to be shifted out via the JTAG port.

57.4.1 JTAG output data register (JOUT)

The JOUT register captures data from the JOUT_IPS register upon execution of the JOUT_READ JTAG instruction. It also holds the JIN_RDY and JOUT_RDY status bits. This register is reset by system power-on reset and JTAG reset. The following figure shows the format of the JOUT register.



1. The reset value of the JOUT register is 34'b0.

The JOUT register is described in the following table.

Table 57-6. JOUT JTAG register field descriptions

Field	Description
JOUT_IPS Data	Data value from JOUT_IPS register
JIN_RDY	State of JIN_RDY bit from MSR
JOUT_RDY	State of JOUT_RDY bit from MSR

57.4.2 JTAG input data register (JIN)

Data is written to the JIN register via JTAG during execution of the JIN_WRITE JTAG instruction. The JIN data is later captured in the JIN_IPS register to be read via IPS. This register is reset by system power-on reset and JTAG reset. The following figure shows the format of the JIN register.



1. The reset value of the JIN register is 32'b0.

The JIN register is described in the following table.

Table 57-7. JIN JTAG register field descriptions

Field	Description
JIN Data	Contains data to be captured in JIN_IPS register upon exit of Update-DR state when executing WRITE_JIN JTAG instruction.

57.5 Functional description

The JDC module provides the ability to shift in data via the JTAG port and capture that data into a memory mapped register that can be accessed via IPS. It also provides the ability to capture data written to a memory mapped register into a JTAG shift register for output via the JTAG port. An overview of the module functionality is described below.

- A software write to the JOUT_IPS register sets the JOUT_RDY flag bit, indicating new data is available to be read from the JOUT register via the JTAG port. The state of the JOUT_RDY bit is reflected in the MSR and also ported out to the JOUT register. A JTAG read of the JOUT register via execution of the JOUT_READ

instruction with a JOUT_RDY bit whose value is logic 1 indicates the register contains new data. The JOUT_RDY flag bit is cleared upon exit of the Capture-DR JTAG state during execution of the JOUT_READ instruction. Clearing the JOUT_RDY bit indicates to software that a new data value can be written to the JOUT_IPS register.

- A JTAG write to the JIN register via execution of the JIN_READ JTAG instruction updates the contents of the JIN_IPS register upon exit of the Update-DR state. An update of the JIN_IPS register sets the JIN_RDY flag bit, indicating new data is available to be read via IPS. The state of the JIN_RDY bit is reflected in the MSR register and also ported out to the JOUT register. The JIN_RDY flag bit is cleared upon software read of the JIN_IPS register. A JTAG read of the JOUT register with a JIN_RDY value of logic 0 indicates new data can be written to the JIN register.

57.5.1 JTAG register access

Refer to the JTAGC documentation for information on how to access the JTAG registers.

57.5.1.1 JDC block instructions

The JDC block implements the IEEE 1149.1-2001 defined instructions listed in the following table. This section gives an overview of each instruction; refer to the IEEE 1149.1-2001 standard for more details. All undefined opcodes are reserved.

Table 57-8. JTAG instructions

Instruction	Code[4:0]	Instruction Summary
Reserved	00001	Factory debug reserved
JOUT_READ	00010	Selects JOUT data register. Data from JOUT_IPS is captured into JOUT data register upon entry to Capture-DR state while JOUT_READ is active.
JIN_WRITE	01110	Selects JIN data register. Data from JIN is captured into JIN_IPS upon exit of Update-DR state while JIN_WRITE is active.
BYPASS	11111	Selects bypass register for data operations
Reserved	All other opcodes	Decoded to select bypass register

Chapter 58

Nexus Development Interface (NDI)

58.1 Overview

The NDI integrates seven individual Nexus modules that provide the chip's development support interface.

All seven Nexus modules are connected to the Nexus Port Controller (NPC). The following table lists the modules.

Table 58-1. Nexus modules of NDI

Nexus modules	For detailed information, see
Two Nexus e200z7 core interfaces (NZ7C3): one each for Core0 and Core1	<i>e200z759n3 Core Reference Manual</i> (document number e200z759n3CRM)
Two Nexus eTPU development interfaces (NDEDI and NSEDI)	Nexus eTPU Development Interface (NDEDI)
Two Nexus crossbar eDMA interfaces (NXDM): one each for eDMA_A and eDMA_B One Nexus crossbar SIPI interface (NXSI)	Nexus eDMA Interface (NXDM) and Nexus SIPI Interface (NXSI)

The chip has no debug capability for the CSE module. CSE is not traceable.

NOTE

In each core, simultaneous access to debug control registers by software and Nexus hardware is not possible. When debug registers are allocated to Nexus hardware, they cannot be accessed by software.

58.2 Nexus clients for TAP access

After the NDI is out of the reset state: To grant the NDI ownership of the TAP, a specific instruction must be loaded in the JTAG Controller (JTAGC) module. Each Nexus client has its own JTAGC instruction opcode for ownership of the TAP, granting that client read/write access to the TAP's registers. The following table shows the JTAGC instruction opcode for each Nexus client. After the JTAGC opcode for a client has been loaded, the client is enabled by loading its NEXUS-ENABLE instruction.

Table 58-2. JTAG client select instructions

JTAGC instruction	Opcode	Description
ACCESS_AUX_NPC	10000	Enables access to the NPC TAP controller
ACCESS_AUX_ONCE	10001	Enables access to the primary OnCE TAP controller (Core0)
ACCESS_AUX_eTPU	10010	Enables access to the eTPU Nexus TAP controller (ETPU A/B)
ACCESS_AUX_NXDM	10011	Enables access to the eDMA Nexus TAP controller (eDMA_A)
Reserved	10100	Not used
ACCESS_AUX_SIPI	10101	Enables access to the SIPI Nexus TAP controller
ACCESS_AUX_eTPU_Secondary	10110	Enables access to secondary ETPU Nexus TAP controller (ETPU C)
ACCESS_AUX_NXDM_B	10111	Enables access to the second eDMA Nexus TAP controller (eDMA_B)
Reserved	11000	Not used
ACCESS_AUX_ONCE_1	11001	Enables access to the secondary OnCE TAP controller (Core1)
ACCESS_AUX_PAR	11010	Enables parallel access to both OnCE TAP controllers (Core0 and Core1)
ACCESS_AUX_MULTI	11100	Enables serial access to both OnCE TAP controllers (Core0 followed by Core1)
ACCESS_AUX_JDC	11101	Enables access to the JDC TAP controller
Reserved	11110	Not used
BYPASS	11111	Selects bypass register for data operations

58.3 Debug break and cross-triggering

This chip supports central break control among the cores and debug modules such as eTPU, eDMA, and Watchdog, allowing synchronous restart control for all cores when exiting debug mode.

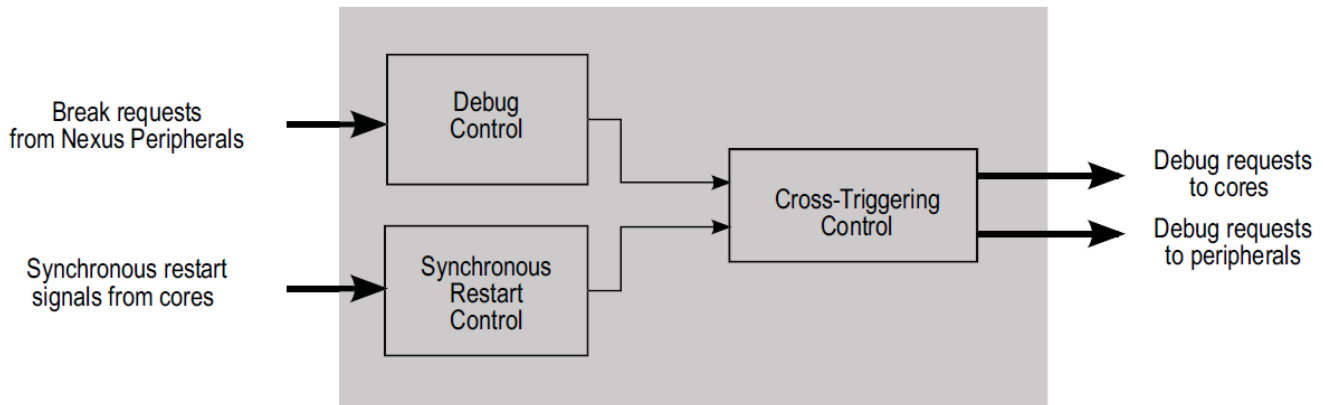


Figure 58-1. Debug break and cross-triggering logic

58.3.1 Break control

Multiple modules on the chip, including the Nexus trace blocks, have debug features that allow them to request device-wide breaks to enter debug mode. All debug mode entry requests are combined (logically ORed) together, and the result produces the debug requests to the cores and peripherals.

The cross-triggering logic generates these debug request signals:

- Debug Request signal (active high) for all non-core peripherals
- External Debug Request (active high) for requesting an external debug event to cores

58.3.2 Cross-triggering control

Cross-triggering provides the ability to select whether modules enter debug mode upon receiving halt request signals from Nexus modules. In response to debug mode entry, some modules can be programmed to halt operation or to continue normal operation. This cross-triggering feature is implemented through debug control bits in the modules. The cores and peripherals can be programmed selectively with their respective control bits.

58.3.3 Synchronous restart control

The cross-triggering logic implements a feature that enables all debug peripherals—cores and peripherals that are put in the debug state—to restart synchronously, as follows:

1. The debugging tool sequentially scans the CPUSCR for each core to restore the proper processor state.
2. The debugging tool scans into the main chip TAP controller the instruction whose execution selects all core TAP controllers in parallel.
3. To enable any combination of cores to exit debug mode synchronously: The debugging tool simultaneously scans into each core's TAP instruction register (OCMD) the 10-bit sequence 180h, which sets both the GO and EX fields to 1 and clears RS[0:6] field to 0.
4. If a core exits debug mode, it sets a flag OnCE go+exit command.
5. When the OR of the core OnCE go+exit command signals is asserted, all other selected non-cores and peripherals in debug synchronously exit debug mode.

58.4 Nexus eDMA Interface (NXDM) and Nexus SIPI Interface (NXSI)

The e200z7 eDMA Nexus module (NXDM) and SIPI module (NXSI) comply with the Class 3 defined data trace feature of the IEEE-ISTO 5001-2003 standard.

58.4.1 Introduction

The NXDM can be programmed to trace data accesses for the eDMA module on the system bus. The NXSI can be programmed to trace data accesses for the SIPI module on the system bus. This eDMA module and SIPI module as well as the Nexus module are components of the e200z7 platform. All output messages and register accesses are compliant with the protocol defined in the IEEE-ISTO 5001 standard.

NOTE

The auxiliary port and its signals, such as MCKO, MSEO[1:0], MDO[15:0] and others, are referenced. The device NPC module arbitrates the access of the single auxiliary port. The functions of the NXDM and SIPI modules are described without the interaction of the NPC, as if Nexus has a dedicated auxiliary port, to simplify the description.

58.4.1.1 Block Diagrams

Figure 58-2 shows a block diagram of the NXDM. The block diagram of the NXSI is the same.

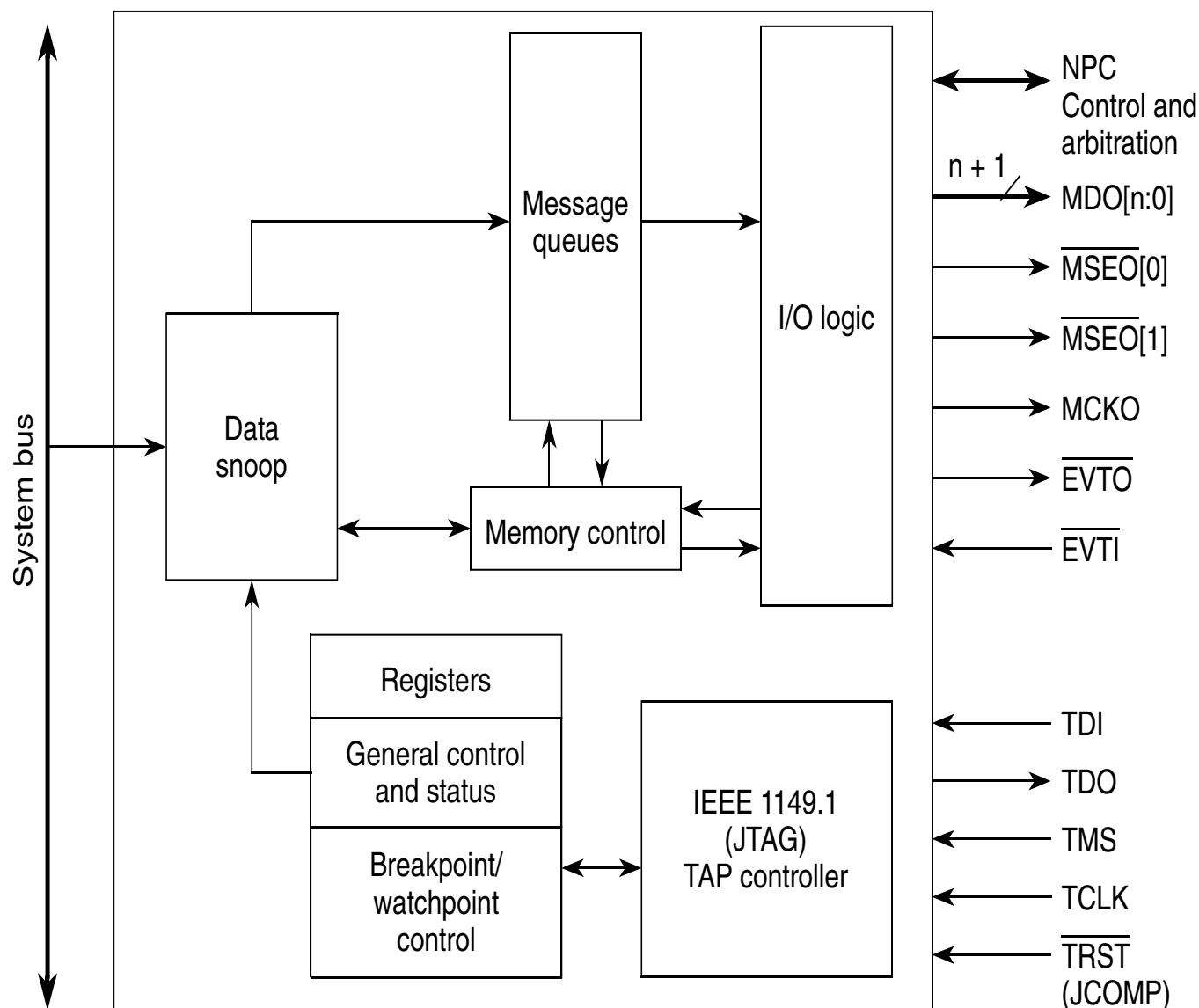


Figure 58-2. NXDM and NXSI Block Diagram

58.4.1.2 Features

Features include the following:

- Data trace via data write messaging (DWM) and data read messaging (DRM). This provides the capability for the development tool to trace reads and/or writes through the eDMA and SIPI modules to (selected) internal memory resources.
- Watchpoint messaging via the auxiliary pins.
- Watchpoint trigger enable of data trace messaging (DTM).

- Registers for data trace, watchpoint generation, and watchpoint trigger.
- All features controllable and configurable via the JTAG port.
- Power management.
 - Low power design
 - Dynamic power management of FIFOs and control logic

58.4.2 External Signal Description

The NXDM and NXSI modules use the same pins and pin protocol as the JTAG Controller and NPC modules.

58.4.2.1 Rules for Output Messages

The NXDM and NXSI modules observe the same rules for output messages as the NPC.

58.4.2.2 Auxiliary Port Arbitration

The NXDM and NXSI modules arbitrate for the shared Nexus port. This arbitration is handled by the NPC based on prioritized requests from the NXDM, NXSI, and the other Nexus clients sharing the port.

58.4.3 NXDM and NXSI Programmers Model

This section describes the programmers model. Nexus registers are accessed using the JTAG port in compliance with IEEE 1149.1. See the JTAGC module details for more information about Nexus register access.

58.4.3.1 NXDM and NXSI Nexus Register Map

Table 58-3. NXDM and NXSI Register Map

Nexus Register	Nexus Access Opcode	Read/Write	Read Address	Write Address
Client Select Control (CSC) ¹	0x1	R	0x02	-

Table continues on the next page...

Table 58-3. NXDM and NXSI Register Map (continued)

Nexus Register	Nexus Access Opcode	Read/Write	Read Address	Write Address
Port Configuration Register (PCR) ¹	Refer to NPC	R/W	-	-
Development Control 1 (DC1_n)	0x2	R/W	0x04	0x05
Development Control 2 (DC2_n)	0x3	R/W	0x05	0x06
Watchpoint Trigger (WT_n)	0xB	R/W	0x16	0x17
Data Trace Control (DTC_n)	0xD	R/W	0x1A	0x1B
Data Trace Start Address 1 (DTSA1_n)	0xE	R/W	0x1C	0x1D
Data Trace Start Address 2 (DTSA2_n)	0xF	R/W	0x1E	0x1F
Data Trace End Address 1 (DTEA1_n)	0x12	R/W	0x24	0x25
Data Trace End Address 2 (DTEA2_n)	0x13	R/W	0x26	0x27
Breakpoint/Watchpoint Control Register 1 (BWC1_n)	0x16	R/W	0x2C	0x2D
Breakpoint/Watchpoint Control Register 2 (BWC2_n)	0x17	R/W	0x2E	0x2F
Breakpoint/Watchpoint Address Register 1 (BWA1_n)	0x1E	R/W	0x3C	0x3D
Breakpoint/Watchpoint Address Register 2 (BWA2_n)	0x1F	R/W	0x3E	0x3F
Reserved	0x20-0x3F	-	0x40-0x7E	0x41-0x7F

1. The CSC and PCR registers are shown in this table as part of the Nexus programmer's model. They are only present at the top level Nexus3 controller (NPC), not in the NXDM or NXSI module. The device's CSC register is readable through Nexus3; the PCR is shown for reference only.

58.4.3.2 NXDM and NXSI Registers

Detailed register definitions for the NXDM and NXSI implementation are as follows:

58.4.3.2.1 Development Control Register DC1

The development control registers control the basic development features of the NXDM and NXSI modules.

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	OPC	MCK_DIV				0	0	WEN	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	C	0	0	0	0	0	0	0	0	0					
W												EIC			TM	
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 58-3. Development Control Register 1 (DC1)

Table 58-4. DC1 Field Description

Field	Description
31 OPC ¹	Output port mode control. 0 Reduced port mode configuration 1 Full port mode configuration
30-29 MCK_DIV ¹	Nexus message clock divide ratio. 00 MCKO is 1x system bus clock frequency 01 MCKO is 1/2x system bus clock frequency 10 MCKO is 1/4x system bus clock frequency 11 MCKO is 1/8x system bus clock frequency
28-27 EOC	EVT $\overline{O}$ control. 00 $\overline{EVT\overline{O}}$ upon occurrence of watchpoint (internal or external) 01 $\overline{EVT\overline{O}}$ upon entry into system-level debug mode (ipg_debug) 1X Invalid value
26-25	Reserved, read as 0.
24 WEN	Watchpoint trace enable. 0 Watchpoint messaging disabled 1 Watchpoint messaging enabled
23-5	Reserved, read as 0.
4-3 EIC	EVTI control. 00 $\overline{EVTI}$ for synchronization (Data Trace) 01 Invalid value 10 $\overline{EVTI}$ disabled for this module 11 Invalid value
2-0 TM	Trace mode. 000 No Trace 1XX Program trace enabled (not supported) ² X1X Data trace enabled XX1 Ownership Trace enable (not supported) ²

1. The output port mode control bit (OPC) and MCKO divide bits (MCK_DIV) are shown for clarity. These functions are controlled globally by the NPC port control register (PCR).
2. The XBAR bus Nexus trace module only supports data trace and watchpoint trace, and does not support program or ownership trace.

58.4.3.2.2 Development Control Register 2 (DC2)

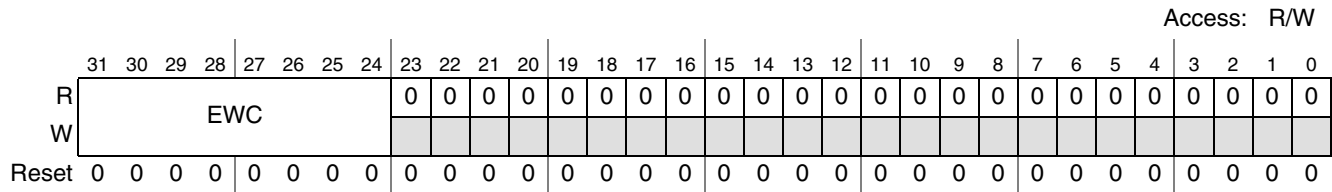


Figure 58-4. Development Control Register 2 (DC2)

Table 58-5. DC2 Field Description

Field	Description
31-24 EWC ¹	EVTO Watchpoint Configuration 00000000 = No watchpoints trigger $\overline{\text{EVTO}}$ 1XXXXXXX = Invalid value X1XXXXXX = Invalid value XX1XXXXX = Invalid value XXX1XXXX = Invalid value XXXX1XXX = Internal watchpoint #1 triggers EVTO XXXXX1XX = Internal watchpoint #2 triggers EVTO XXXXXX1X = Invalid value XXXXXXXX1 = Invalid value
23-0	Reserved, read as 0.

1. The EOC bits in DC1 must be programmed to trigger $\overline{\text{EVTO}}$ on watchpoint occurrence for the EWC bits to have any effect.

58.4.3.2.3 Watchpoint Trigger Register (WT)

The watchpoint trigger register allows the watchpoints defined internally to the NXDM and NXSI modules to trigger actions. These watchpoints can control data trace enable and disable. The WT bits can be used to produce an address related window for triggering trace messages.

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	0	0	0	DTS				DTE				0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 58-5. Watchpoint Trigger Register (WT)

Table 58-6. WT Field Description

Field	Description
31-26	Reserved, read as 0.
25-23 DTS	DTS - Data trace start control 000 Trigger disabled 001-100 Invalid value 101 Use internal watchpoint #1 (BWA1 register) 110 Use internal watchpoint #2 (BWA2 register) 111 Invalid value
22-20 DTE	DTE - Data trace end control 000 Trigger disabled 001-100 Invalid value 101 Use internal watchpoint #1 (BWA1 register) 110 Use internal watchpoint #2 (BWA2 register) 111 Invalid value
19-0	Reserved, read as 0.

NOTE

The WT bits ONLY enable data trace if the tm bits within the development control register (DC) have not already been set to enable data trace.

58.4.3.2.4 Data Trace Control Register (DTC)

The data trace control register controls whether DTM Messages are restricted to reads, writes or both for a user programmable address range. There are two data trace channels controlled by the DTC for the NXDM and NXSI modules.

Access: R/W

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	RWT1		RWT2		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	RC1		RC2		0	0	0	0
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 58-6. Data Trace Control Register (DTC)**Table 58-7. DTC Field Description**

Bit	Description
31-30 RWT1	Read/write trace 1 00 No trace messages generated X1 Enable data read trace 1X Enable data write trace
29-28 RWT2	Read/write trace 2 00 No trace messages generated X1 Enable data read trace 1X Enable data write trace
27-8	Reserved, read as 0.
7 RC1	Range control 1 0 Condition trace on address within range (endpoints inclusive) 1 Condition trace on address outside of range (endpoints exclusive)
6 RC2	Range control 2 0 Condition trace on address within range (endpoints inclusive) 1 Condition trace on address outside of range (endpoints exclusive)
5-0	Reserved, read as 0.

58.4.3.2.5 Data Trace Start Address Registers 1 and 2 (DTSA1 and DTSA2)

The data trace start address registers define the start addresses for each trace channel.

Access: R/W

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	DATA TRACE START ADDRESS																															
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 58-7. Data Trace Start Address Registers (DTSA1, DTSA2)

58.4.3.2.6 Data Trace End Address Registers 1 and 2 (DTEA1 and DTEA2)

The data trace end address registers define the end addresses for each trace channel.

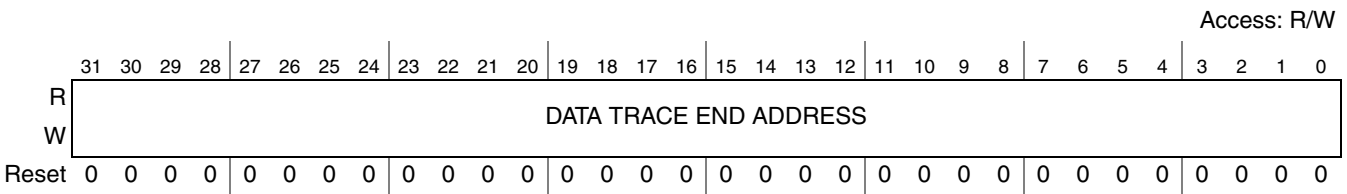


Figure 58-8. Data Trace Start Address Registers (DTEA1, DTEA2)

Table 58-8 illustrates the range that is selected for data trace for various cases of DTSA being less than, greater than, or equal to DTEA.

Table 58-8. Data Trace Address Range Options

Programmed Values	Range Control Bit Value	Range Selected
DTSA < or = DTEA	0	DTSA→ ←DTEA
DTSA < or = DTEA	1	← DTSA DTEA →
DTSA > DTEA	-	Invalid range, no trace

NOTE

DTSA must be less than (or equal to) DTEA to guarantee correct data write/read traces. When the range control bit is 0 (internal range), accesses to DTSA and DTEA addresses are traced. When the range control bit is 1 (external range), accesses to DTSA and DTEA are not traced.

58.4.3.2.7 Breakpoint / Watchpoint Control Register 1 (BWC1)

Breakpoint/watchpoint control register 1 controls attributes for generation of NXDM and NXSI watchpoint number 1.

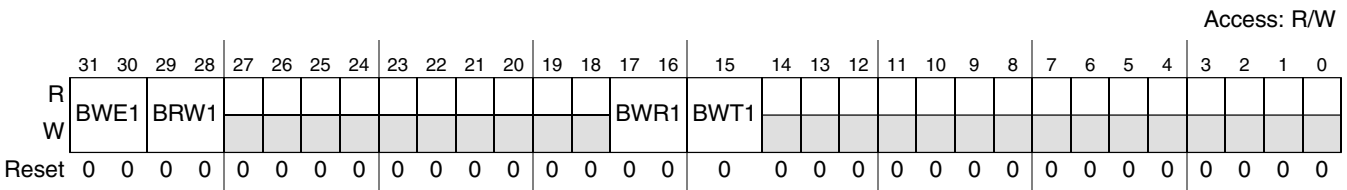


Figure 58-9. Break / Watchpoint Control Register 1 (BWC1)

Table 58-9. BWC1 Field Description

Field	Description
31-30 BWE1	Breakpoint/watchpoint #1 enable 00 Internal Nexus watchpoint #1 disabled 01-10 Invalid value 11 Internal Nexus watchpoint #1 enabled
29-28 BRW1	Breakpoint/watchpoint #1 read/write select 00 Watchpoint #1 hit on read accesses 01 Watchpoint #1 hit on write accesses 10 Watchpoint #1 on read or write accesses 11 Invalid value
27-18	Reserved, read as 0.
17-16 BWR1	Breakpoint/watchpoint #1 register compare 00 No register compare (same as BWC1[31:30] = 2'b00) 01 Invalid value 10 Compare with BWA1 value 11 Invalid value
15 BWT1	Breakpoint/watchpoint #1 type 0 Invalid value 1 Watchpoint #1 on data accesses
14-0	Reserved, read as 0.

58.4.3.2.8 Breakpoint / Watchpoint Control Register 2 (BWC2)

Breakpoint/watchpoint control register2 controls attributes for generation of NXDM and NXSI watchpoint number 2.

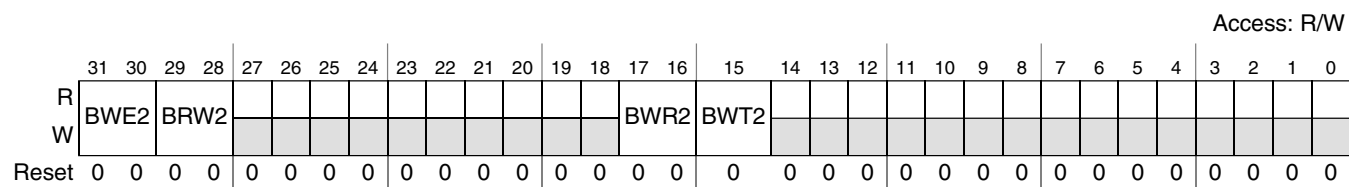
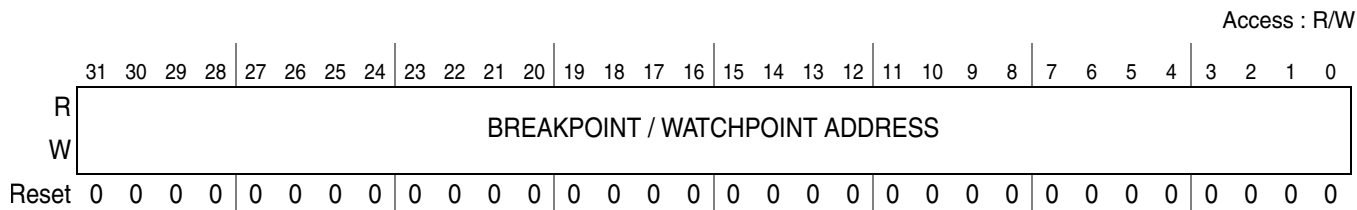
**Figure 58-10. Break / Watchpoint Control Register 2 (BWC2)**

Table 58-10. BWC1 Field Description

Field	Description
31-30 BWE2	Breakpoint/watchpoint #2 enable 00 Internal Nexus watchpoint #2 disabled 01-10 Invalid value 11 Internal Nexus watchpoint #2 enabled
29-28 BRW2	Breakpoint/watchpoint #1 read/write select 00 Watchpoint #2 hit on read accesses 01 Watchpoint #2 hit on write accesses 10 Watchpoint #2 on read or write accesses 11 Invalid value
27-18	Reserved, read as 0.
17-16 BWR2	Breakpoint/watchpoint #2 register compare 00 No register compare (same as BWC1[31:30] = 2'b00) 01 Invalid value 10 Compare with BWA2 value 11 Invalid value
15 BWT2	Breakpoint/watchpoint #2 type 0 Invalid value 1 Watchpoint #2 on data accesses
14-0	Reserved, read as 0.

58.4.3.2.9 Breakpoint/Watchpoint Address Registers 1 and 2 (BWA1 and BWA2)

The breakpoint/watchpoint address registers are compared with bus addresses to generate internal watchpoints.

**Figure 58-11. Breakpoint / Watchpoint Address Registers (BWA1, BWA2)**

58.4.3.2.10 Unimplemented Registers

Unimplemented registers are those with client select and index value combinations other than the ones listed in [Table 58-3](#). For unimplemented registers, the NXDM and NXSI modules drive TDO to zero during the "SHIFT-DR" state. The modules also transmit an error message with the invalid access opcode encoding.

58.4.3.2.11 Programming Considerations ($\overline{\text{RESET}}$)

If Nexus3 register configuration is to occur during system reset (as opposed to debug mode), all NXDM configuration should be completed between the negation of JCOMP and system reset de-assertion, after the JTAG DID register has been read by the tool.

58.4.3.2.12 IEEE 1149.1 (JTAG) Test Access Port

The NXDM and NXSI modules use the IEEE 1149.1 TAP controller for accessing Nexus resources. The JTAG signals themselves are shared by all TAP controllers on the device. See the JTAGC module details for more information about the JTAG interface.

The NXDM and NXSI modules implement a 4-bit instruction register (IR). The valid instructions and method for register access are outlined in the JTAGC module details.

58.4.3.2.12.1 NXDM and NXSI JTAG DID Register

This JTAG DID register that is included in the NXDM and NXSI modules provides key development attributes to the development tool concerning the NXDM and NXSI blocks. The register is accessed through the standard JTAG IR/DR paths. Refer to the PMC chapter.

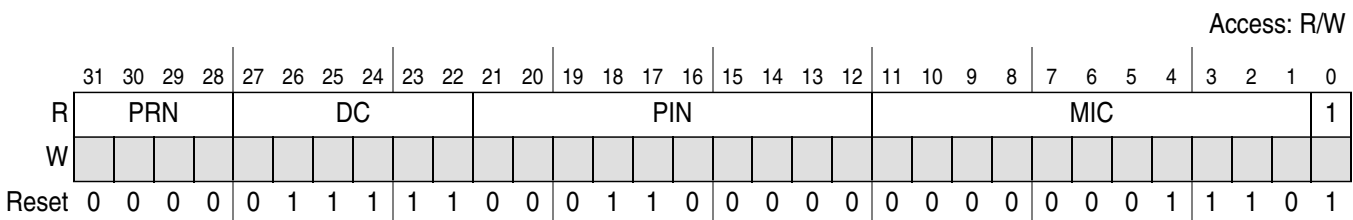


Figure 58-12. NXDM and NXSI JTAG DID Register

Table 58-11. NXDM and NXSI JTAG DID Field Descriptions

Field	Description
31-28 PRN ¹	Embedded part revision number (0x0)
27-22 DC	Freescale design center ID number (0x1F)

Table continues on the next page...

Table 58-11. NXDM and NXSI JTAG DID Field Descriptions (continued)

Field	Description
21-12 PIN	NXDM and NXSI module part identification number, defines the features set. (0x60)
11-1 MIC	Manufacturer identity code 0x00E Freescale
0	Fixed per JTAG 1149.1 1 Always set

1. The revision number is initially 0 and could change in the future.

58.4.3.2.12.2 Enabling the NXDM and NXSI TAP Controllers

Assertion of a power-on-reset signal or assertion of the JCOMP pin resets all TAP controllers on the device. Upon exit from the test-logic-reset state, the IR value is loaded with the JTAG DID. When the NXDM or NXSI TAP is accessed, this information helps the development tool obtain information about the Nexus module it is accessing, such as version, sequence, feature set, and so forth.

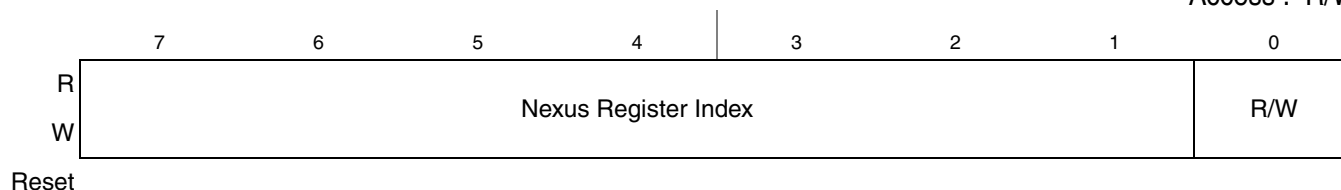
58.4.3.2.12.3 NXDM and NXSI Register Access via JTAG

Access to Nexus register resources is enabled by loading a single instruction (NEXUS_ACCESS) into the JTAG Instruction Register (IR). This IR is part of the IEEE 1149.1 TAP controller within the NXDM and NXSI modules.

After the JTAG NEXUS_ACCESS instruction has been loaded, the JTAG port allows tool/target communications with all Nexus registers according to the map in [Table 58-3](#).

Reading/writing of a Nexus register then requires two (2) passes through the data-scan (DR) path of the JTAG state machine.

1. The first pass through the DR selects the Nexus register to be accessed by providing an index (refer to [Table 58-3](#)), and the direction (read/write). This is achieved by loading an 8-bit value into the JTAG data register (DR). [Figure 58-13](#) shows the format of this register.
2. The second pass through the DR then shifts the data in or out of the JTAG port, lsb first.
 - a. During a read access, data is latched from the selected Nexus register when the JTAG state machine passes through the capture-DR state.
 - b. During a write access, data is latched into the selected Nexus register when the JTAG state machine passes through the update-DR state.

**Figure 58-13. JTAG DR for NEXUS Register Access****Table 58-12. DR Read/Write Encoding**

Nexus Register Index	Description
Read/Write (R/W)	0 Read 1 Write

58.4.4 Functional Description

58.4.4.1 Enabling NXDM and NXSI Operation

The NXDM (and NXSI) module is enabled by loading a single instruction (ACCESS_AUX_TAP_DMA or ACCESS_AUX_TAP_NXSI) into the JTAG instruction register (IR), and then loading the corresponding OnCE OCMD register with the NEXUS_ACCESS instruction. After it is enabled, the module is ready to accept control input via the JTAG pins.

The Nexus module is disabled when the JTAG state machine reaches the test-logic-reset state. This state can be reached by the assertion of the JCOMP pin or by cycling through the state machine using the TMS pin. The Nexus module is also disabled if a power-on reset (POR) event occurs.

If the NXDM (and NXSI) module is disabled, no trace output is provided, and the module disables (drive inactive) auxiliary port output pins (MDO[15:0], MSEO[1:0], MCKO). Nexus registers are not be available for reads or writes.

58.4.4.2 TCODEs Supported by NXDM and NXSI

The NXDM and NXSI pins allow for flexible transfer operations via public messages. A TCODE defines the transfer format, the number and/or size of the packets to be transferred, and the purpose of each packet. The IEEE-ISTO 5001-2003 standard defines a set of public messages. The NXDM and NXSI blocks currently support the public TCODEs seen in [Table 58-13](#).

Table 58-13. Public TCODEs Supported

Message Name	Packet Size Bits		Packet Name	Packet Type	Packet Description
	Min	Max			
Data Trace - Data Write Message	6	6	TCODE	Fixed	TCODE number = 5
	4	4	SRC	Fixed	Source processor identifier (multiple Nexus configuration)
	3	3	DSZ	Fixed	Data size (refer to Table 58-15)
	1	32	U-ADDR	Variable	Unique portion of the data write value
	1	64	DATA	Variable	Data write value
Data Trace - Data Read Message	6	6	TCODE	Fixed	TCODE number = 6
	4	4	SRC	Fixed	Source processor identifier (multiple Nexus configuration)
	3	3	DSZ	Fixed	Data size (refer to Table 58-15)
	1	32	U-ADDR	Variable	Unique portion of the data read value
	1	64	DATA	Variable	Data read value
Error Message	6	6	TCODE	Fixed	TCODE number = 8
	4	4	SRC	Fixed	Source processor identifier (multiple Nexus configuration)
	5	5	ECODE	Fixed	Error code (refer to Table 58-14)
Data Trace - Data Write Message w/ Sync	6	6	TCODE	Fixed	TCODE number = 13 (0xD)
	4	4	SRC	Fixed	Source processor identifier (multiple Nexus configuration)
	3	3	DSZ	Fixed	Data size (refer to Table 58-15)
	1	32	F-ADDR	Variable	Full access address (leading zero (0) truncated)
	1	64	DATA	Variable	Data write value
Data Trace - Data Read Message w/ Sync	6	6	TCODE	Fixed	TCODE number = 14 (0xE)
	4	4	SRC	Fixed	Source processor identifier (multiple Nexus configuration)
	3	3	DSZ	Fixed	Data size (refer to Table 58-15)
	1	32	F-ADDR	Variable	Full access address (leading zero (0) truncated)
	1	64	DATA	Variable	Data read value

Table continues on the next page...

Table 58-13. Public TCODEs Supported (continued)

Message Name	Packet Size Bits		Packet Name	Packet Type	Packet Description
	Min	Max			
Watchpoint Message	6	6	TCODE	Fixed	TCODE number = 15 (0xF)
	4	4	SRC	Fixed	Source processor identifier (multiple Nexus configuration)
	4	4	WPHIT	Fixed	Number indicating watchpoint sources

Table 58-14. Error Code (ECODE) Encoding (TCODE = 8)

Error Code (ECODE)	Description
00000	Invalid value
00001	Invalid value
00010	Data Trace overrun
00011	Invalid value
00100	Invalid value
00101	Invalid access opcode (Nexus Register unimplemented)
00110	Watchpoint overrun
00111	Invalid value
01000	Data Trace and Watchpoint overrun
01001-11111	Invalid value

Table 58-15. Data Trace Size (DSZ) Encodings (TCODE = 5, 6, 13, 14)

DTM Size Encoding	Transfer Size
000	Byte
001	Halfword (two bytes)
010	Word (four bytes)
011	Doubleword (eight bytes)
100-111	Invalid value

58.4.4.2.1 Data Trace

This section deals with the data trace mechanism supported by the NXDM and NXSI modules. Data trace is implemented via data write messaging (DWM) and data read messaging (DRM).

58.4.4.2.2 Data Trace Messaging (DTM)

XDM and NXSI data trace messaging is accomplished by snooping the NXDM and NXSI data bus, and storing the information for qualifying accesses (based on enabled features and matching target addresses). The NXDM (and NXSI) module traces all data access that meet the selected range and attributes.

NOTE

Data trace is ONLY performed on DMA or SIPI accesses to the system bus.

58.4.4.2.3 DTM Message Formats

The NXDM (and NXSI) block supports five types of DTM Messages - data write, data read, data write synchronization, data read synchronization and error messages.

58.4.4.2.3.1 Data Write and Data Read Messages

The data write and data read messages contain the data write/read value and the address of the write/read access, relative to the previous data trace message. Data write message and data read message information is messaged out in the following format:

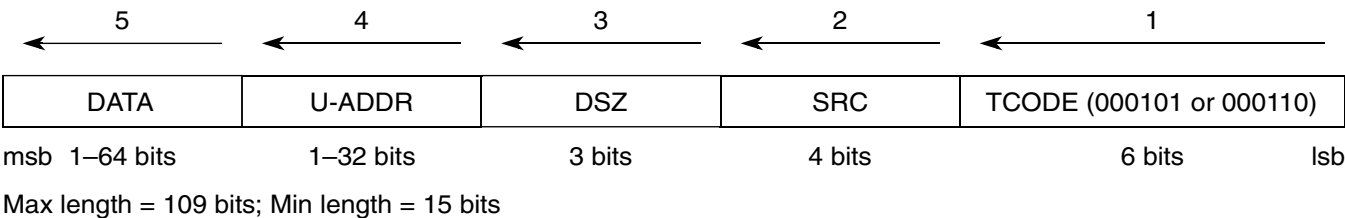


Figure 58-14. Data Write/Read Message Format

58.4.4.2.3.2 DTM Overflow Error Messages

An error message occurs when a new message cannot be queued due to the message queue being full. The FIFO discards incoming messages until it has completely emptied the queue. After it is emptied, an error message is queued. The error encoding indicates which types of messages attempted to be queued while the FIFO was being emptied.

If only a data trace message attempts to enter the queue while it is being emptied, the error message incorporates the data trace only error encoding (00010). If a watchpoint also attempts to be queued while the FIFO is being emptied, then the error message incorporates error encoding (01000).

Error information is messaged out in the following format:

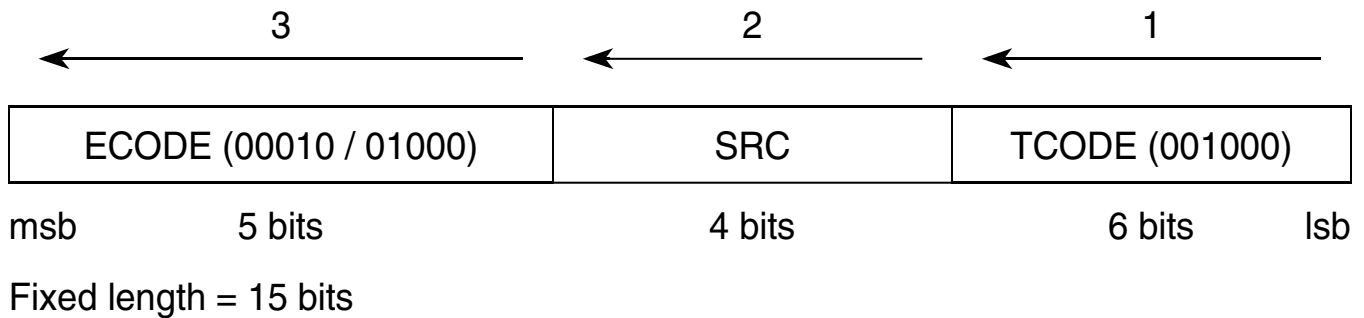


Figure 58-15. Error Message Format

58.4.4.2.3.3 Data Trace Synchronization Messages

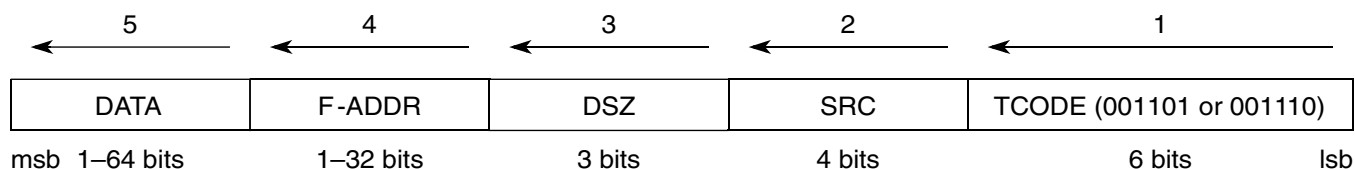
A data trace write/read w/ sync. message is messaged via the auxiliary port (provided data trace is enabled) for the following conditions (refer to [Table 58-16](#)):

- Initial data trace message upon exit from system reset or whenever data trace is enabled is a synchronization message.
- Upon returning from a low power state, the first data trace message is a synchronization message.
- Upon returning from debug mode, the first data trace message is a synchronization message.
- After occurrence of queue overrun (can be caused by any trace message), the first data trace message is a synchronization message.
- After the periodic data trace counter has expired indicating 255 without-sync data trace messages have occurred since the last with-sync message occurred.
- Upon assertion of the Event In ($\overline{\text{EVTI}}$) pin, the first data trace message is a synchronization message if the eic bits of the dc register have enabled this feature.
- Upon data trace write/read after the previous dtm message was lost due to an attempted access to a secure memory location.
- Upon data trace write/read after the previous dtm message was lost due to a collision entering the fifo between the dtm message and any of the following: error message, or watchpoint message.

Data trace synchronization messages provide the full address (without leading zeros) and insure that development tools fully synchronize with data trace regularly.

Synchronization messages provide a reference address for subsequent DTMs, in which only the unique portion of the data trace address is transmitted. The format for data trace write/read w/ sync. messages is as follows:

Nexus eDMA Interface (NXDM) and Nexus SIPI Interface (NXSI)



Max length = 109 bits; Min length = 15 bits

Figure 58-16. Data Write/Read w/ Sync Message Format

Exception conditions that result in data trace synchronization are summarized in [Table 58-16.](#)"

Table 58-16. Data Trace Exception Summary

Exception Condition	Exception Handling
System Reset Negation	At the negation of JTAG reset (JCOMP), queue pointers, counters, state machines, and registers within the NXDM and NXSI module are reset. If data trace is enabled, the first data trace message is a data write/read w/ sync. message.
Data Trace Enabled	The first data trace message (after data trace has been enabled) is a synchronization message.
Exit from Low Power/Debug	Upon exit from a low power mode or debug mode the next data trace message is converted to a data write/read w/ sync. message.
Queue Overrun	An error message occurs when a new message cannot be queued due to the message queue being full. The FIFO discards messages until it has completely emptied the queue. After it is emptied, an error message is queued. The error encoding indicates the types of messages that attempted to be queued while the FIFO was being emptied. The next DTM message in the queue is a data write/read w/ sync. message.
Periodic Data Trace Synchronization	A forced synchronization occurs periodically after 255 data trace messages have been queued. A data write/read w/ sync. message is queued. The periodic data trace message counter then resets.
Event In	If the nexus module is enabled, an EVTI assertion initiates a data trace write/read w/ sync. message upon the next data write/read (if data trace is enabled and the eic bits of the dc register have enabled this feature).
Attempted Access to Secure Memory	Any attempted read or write to secure memory locations temporarily disable data trace & cause the corresponding DTM to be lost. A subsequent read/write queues a data trace read/write with sync. message.
Collision Priority	All messages have the following priority: Error → WPM → DTM. A DTM message which attempts to enter the queue at the same time as an error message, or watchpoint message is lost. A subsequent read/write queues a data trace read/write with sync. message.

58.4.4.2.4 DTM Operation

58.4.4.2.4.1 Enabling Data Trace Messaging

Data trace messaging can be enabled in one of two ways.

- Setting the DC1[TM] field to enable data trace
- Using the WT[DTS] field to enable data trace on watchpoint hits

58.4.4.2.4.2 DTM Queueing

NXDM and NXSI implements a programmable depth queue for queuing all messages. Messages that enter the queue are transmitted via the auxiliary pins in the order in which they are queued.

NOTE

If multiple trace messages must be queued at the same time, watchpoint messages have the highest priority (WPM → DTM).

58.4.4.2.4.3 Relative Addressing

The relative address feature is compliant with IEEE-ISTO Nexus 5001-2003 and is designed to reduce the number of bits transmitted for addresses of data trace messages. Relative addressing is the same as described for the NZ7C3.

58.4.4.2.4.4 Data Trace Windowing

Data write/read messages are enabled via the RWT1(2) field in the data trace control register (DTC) for each DTM channel. Data trace windowing is achieved via the address range defined by the DTEA and DTSA registers and by the RC1(2) field in the DTC. All eDMA or SIPI initiated read/write accesses that fall inside or outside these address ranges, as programmed, are candidates to be traced.

58.4.4.2.4.5 System Bus Cycle Special Cases

Table 58-17. System Bus Cycle Special Cases

Special Case	Action
System bus cycle aborted (DABORT asserted)	Cycle ignored
System bus cycle with data error	Data Trace Message discarded
System bus cycle completed without error	Cycle captured and transmitted
System bus cycle is an instruction fetch	Cycle ignored

58.4.4.2.5 Data Trace Timing Diagrams (Eight MDO configuration)

Data trace timing for the NXDM and NXSI is the same as for the NZ7C3.

58.4.4.3 Watchpoint Support

The NXDM and NXSI module provides watchpoint messaging via the auxiliary pins, as defined by IEEE-ISTO 5001-2003.

Watchpoint messages can be generated using the NXDM and NXSI defined internal watchpoints.

58.4.4.3.1 Watchpoint Messaging

Enabling watchpoint messaging is accomplished by setting the watchpoint messaging enable bit, DC1[WEN]. Using the BWC1 and BWC2 registers, two independently controlled internal watchpoints can be initialized. When a DMA or SIPI access address matches on BWA1 or BWA2, a watchpoint message is transmitted.

The Nexus module provides watchpoint messaging using the TCODE. When either of the two possible watchpoint sources asserts, a message is sent to the queue to be messaged out. This message indicates the watchpoint number.

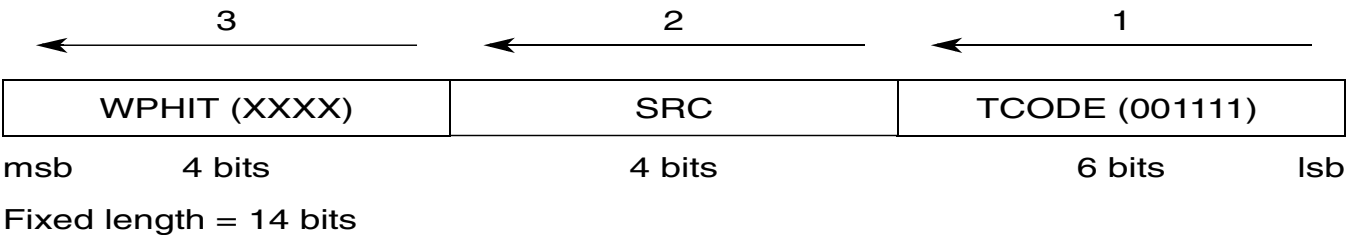


Figure 58-17. Watchpoint Message Format

Table 58-18. Watchpoint Source Description

Watchpoint Source (4 bits)	Watchpoint Description
XXX1	Invalid value
XX1X	Invalid value
X1XX	Internal Watchpoint #1 (BWA1 match)
1XXX	Internal Watchpoint #2 (BWA2 match)

58.4.4.3.2 Watchpoint Error Message

An error message occurs when a new message cannot be queued due to the message queue being full. The FIFO discards messages until it has completely emptied the queue. After it is emptied, an error message is queued. The error encoding indicates which types of messages attempted to be queued while the FIFO was being emptied.

If only a watchpoint message attempts to enter the queue while it is being emptied, the error message incorporates the watchpoint only error encoding (00110). If a data trace message also attempts to enter the queue while it is being emptied, the error message incorporates error encoding (01000).

Error information is messaged out in the following format (refer to [Figure 58-18](#)).

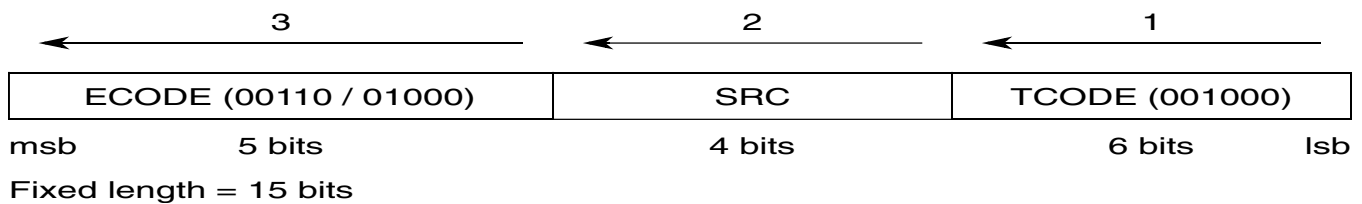


Figure 58-18. Error Message Format

Chapter 59

Nexus Port Controller (NPC)

59.1 Chip-specific NPC information

The NPC connects to several other Nexus modules that compose the chip's Nexus Development Interface (NDI).

For more information, see the chip-level details about debug and development support.

59.1.1 DID register reset value

The reset value of the [Nexus Device ID \(DID\) Register](#) depends on the chip version.

The PRN, PIN, and DC parameters define the chip-specific values of correspondingly named fields in the DID register. The following table provides their values.

Table 59-1. Reset values of PRN, DC, and PIN fields of DID register

DID field	Device-specific value
PRN	0h for masksets 1N45H and 2N45H 1h for maskset 3N45H
DC	20h
PIN	377h

59.1.2 NPC MDO pins

The number of available Message Data Out (MDO) pins varies by NPC mode of operation. The following table shows the number of MDO pins available in full-port mode and reduced-port mode.

Table 59-2. MDO pins available by NPC mode of operation

Mode	Number of available MDO pins
Full-port mode	16
Reduced-port mode	12

59.2 Introduction

The following figure is a block diagram of the Nexus Port Controller (NPC) block.

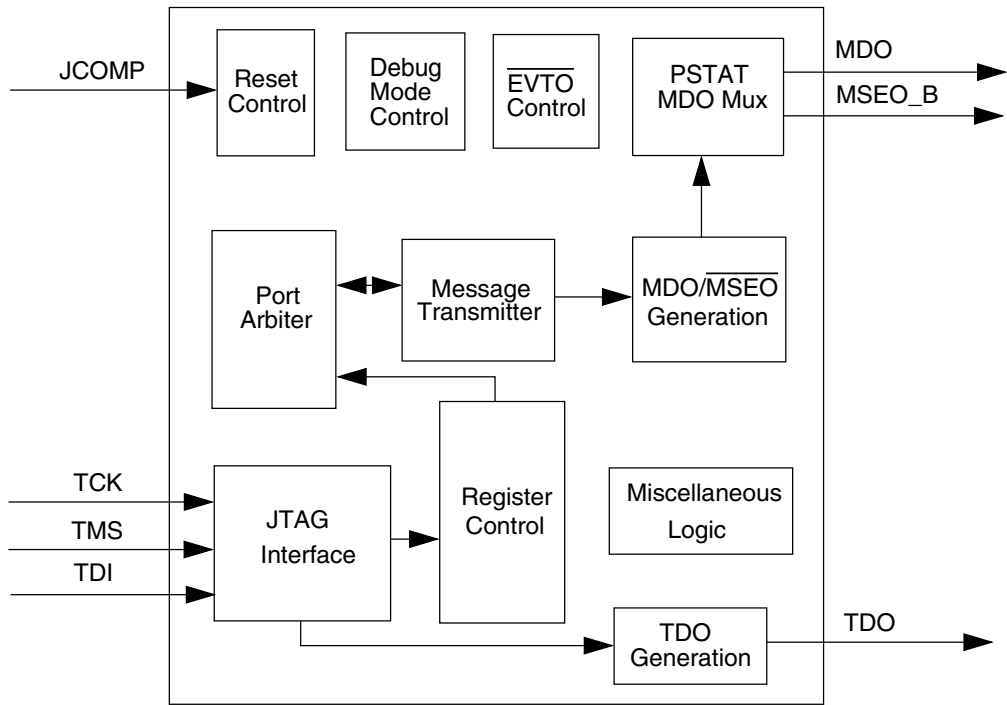


Figure 59-1. Nexus Port Controller Block Diagram

59.2.1 Overview

On a system-on-a-chip device, there are often multiple blocks that require development support. Each of these blocks implements a development interface based on the IEEE-ISTO 5001-2001 standard. The blocks share input and output ports that interface with the development tool. The NPC controls the usage of the input and output port in a manner that allows all the individual development interface blocks to share the port, and appear to the development tool to be a single block.

59.2.2 Features

The NPC block performs the following functions:

- Controls arbitration for ownership of the Nexus Output Port
- Nexus Device Identification Register and Messaging
- Generates MCKO enable and frequency division control signals
- Generates a Nexus DDR mode status indication output signal
- Controls sharing of $\overline{\text{EVTO}}$
- Generates an MCKO clock gating control signal to enable gating of MCKO when the auxiliary output port is idle
- Control of the device-wide debug mode
- Generates asynchronous reset signal for Nexus blocks based on JCOMP input, censorship status, and power-on reset status
- Selectable mode for transmission of processor status information on MDO Auxiliary Output Port pins
- System clock status indication via MDO[0] following power-on reset
- Provides Nexus support for censorship mode

59.2.3 Modes of operation

The NPC block uses the JCOMP input, the censorship status signal and an internal power-on reset indication as its primary reset signals. Upon exit of reset, the mode of operation is determined by the Port Configuration Register (PCR) settings.

59.2.3.1 Reset

The NPC block is asynchronously placed in reset when either power-on reset is asserted, JCOMP is not set for Nexus access, the device enters censored mode, or the TAP controller state machine is in the Test-Logic-Reset state. Holding TMS high for 5 consecutive rising edges of TCK guarantees entry into the Test-Logic-Reset state regardless of the current TAP controller state. Following negation of power-on reset, the NPC remains in reset until the system clock achieves lock. The NPC is unaffected by other sources of reset. While in reset, the following actions occur:

- The TAP controller is forced into the Test-Logic-Reset state
- The output port pins are negated
- The TDI, TMS, and TCK TAP inputs are ignored (when in power-on reset, censored mode or JCOMP not set for NPC operation only)
- Registers default back to their reset values

59.2.3.2 Disabled-Port Mode

In disabled-port mode, output pin port enable signals are negated, thereby disabling message transmission. Any debug feature that generates messages can not be used. The primary features available are class 1 features and read/write access to the registers. Class 1 features include the ability to trigger a breakpoint event indication through EVTO.

59.2.3.3 Full-Port Mode

Full-port mode (FPM) is entered by asserting the MCKO_EN and FPM bits in the PCR. In All trace features are enabled or can be enabled by writing the configuration registers via the TAP. The number of MDO pins available is device-specific.

59.2.3.4 Reduced-Port Mode

Reduced-port mode (RPM) is entered by asserting the MCKO_EN bit and negating the FPM bit in the PCR. All trace features are enabled or can be enabled by writing the configuration registers via the TAP. The number of MDO pins available is device-specific.

59.2.3.5 Censored Mode

In censored mode, it should not be possible to read the contents of internal Flash externally. The NPC and all other Nexus blocks are held in reset when in censored mode , preventing Nexus read/write to memory mapped resources and the transmission of Nexus trace messages.

59.2.3.6 Nexus Double Data Rate Mode

Nexus double data rate (DDR) mode is enabled by asserting the DDR_EN bit in the PCR. DDR mode is only supported for MCKO_DIV settings of sys_clk/2, sys_clk/4 and sys_clk/8. In double data rate mode, message data is updated between each edge (both rising and falling) of MCKO, effectively doubling the message throughput of single data rate mode.

59.3 External signal description

59.3.1 Overview

The NPC pin interface provides for the transmission of messages from Nexus blocks to the external development tools and for access to Nexus client registers. The NPC pin definition is outlined in the following table. Pull information for the JTAG signals can be found in the JTAG chapter.

Table 59-3. NPC Signal Properties

Name	Port	Function	Reset State
EVTO_B	Nexus	Event Out pin	0b1
JCOMP	JTAG	JTAG Compliancy and TAP Sharing Control	—
MDO	Auxiliary	Message Data Out pins	0 ¹
MSEO	Auxiliary	Message Start/End Out pins	0b11
TCK	JTAG	Test Clock Input	—
TDI	JTAG	Test Data Input	—
TDO	JTAG	Test Data Output	High Z ²
TMS	JTAG	Test Mode Select Input	—

1. Following a power-on reset, MDO[0] remains asserted until power-on reset is exited and the system clock achieves lock.
2. TDO output buffer enable is negated when the NPC is not in the Shift-IR or Shift-DR states. A weak pull may be implemented on TDO at the SoC level.

59.3.2 Detailed signal descriptions

This section describes each of the signals listed in [Table 59-3](#) in more detail. Note that the JTAG test clock (TCK) input from the pin is not a direct input to the NPC. The NPC requires two separate input clocks for TCK clocked logic, one for posedge (rising edge TCK) logic and one for negedge (falling edge TCK) logic. Both clocks are derived from the pin TCK, and generated external to the NPC.

59.3.2.1 EVTO_B — Event Out

Event Out ($\overline{\text{EVTO}}$) is an output pin that is asserted upon breakpoint occurrence to provide breakpoint status indication. The $\overline{\text{EVTO}}$ output of the NPC is generated based on the values of the individual $\overline{\text{EVTO}}$ signals from all Nexus blocks that implement the signal.

59.3.2.2 JCOMP — JTAG Compliancy

The JCOMP signal provides the ability to share the TAP. The NPC TAP controller remains in reset until it is enabled using the JCOMP signal.

59.3.2.3 MDO — Message Data Out

Message Data Out (MDO) are output pins used for uploading OTM, BTM, DTM, and other messages to the development tool. The development tool should sample MDO on the rising edge of MCKO. The width of the MDO bus used is determined by reset configuration.

59.3.2.4 MSEO_B — Message Start/End Out

Message Start/End Out ($\overline{\text{MSEO}}$) is an output pin that indicates when a message on the MDO pins has started, when a variable length packet has ended, or when the message has ended. The development tool should sample $\overline{\text{MSEO}}$ on the rising edge of MCKO.

59.3.2.5 TCK — Test Clock Input

Test Clock Input (TCK) pin is used to synchronize the test logic and control register access through the JTAG port.

59.3.2.6 TDI — Test Data Input

Test Data Input (TDI) pin receives serial test instruction and data. TDI is sampled on the rising edge of TCK.

59.3.2.7 TDO — Nexus Test Data Output

Test Data Output (TDO) pin transmits serial output for instructions and data. TDO is three-stateable and is actively driven in the SHIFT-IR and SHIFT-DR controller states. TDO is updated on the falling edge of TCK and sampled by the development tool on the rising edge of TCK.

59.3.2.8 TMS — Test Mode Select

Test Mode Select Input (TMS) pin is used to sequence the IEEE 1149.1-2001 TAP controller state machine. TMS is sampled on the rising edge of TCK.

59.4 Register definition

This section provides a detailed description of the NPC registers accessible to the end user. Individual bit-level descriptions and reset states of the registers are included.

The following table shows the NPC registers by index values. The registers are not memory-mapped and can only be accessed via the TAP. The NPC block does not implement the client select control register because the value does not matter when accessing the registers.

Note that the bypass and instruction registers have no index values. These registers are not accessed in the same manner as Nexus client registers. Refer to the individual register descriptions for more detail.

Table 59-4. NPC registers

Index	Register
0	Device ID Register (DID)
127	Port Configuration Register (PCR)

59.4.1 Register descriptions

This section consists of NPC register descriptions.

59.4.1.1 Bypass register

The bypass register is a single-bit shift register path selected for serial data transfer between TDI and TDO when the BYPASS instruction or any unimplemented instructions are active. After entry into the Capture-DR state, the single-bit shift register is set to a logic 0. Therefore, the first bit shifted out after selecting the bypass register is always a logic 0.

59.4.1.2 Instruction register

The NPC block uses a 4-bit instruction register as shown in the following table. The instruction register is accessed via the SELECT_IR_SCAN path of the tap controller state machine, and allows instructions to be loaded into the block to enable the NPC for register access (NEXUS_ENABLE) or select the bypass register as the shift path from TDI to TDO (BYPASS or unimplemented instructions).

Instructions are shifted in through TDI while the TAP controller is in the Shift-IR state, and latched on the falling edge of TCK in the Update-IR state. The latched instruction value can only be changed in the Update-IR and Test-Logic-Reset TAP controller states. Synchronous entry into the Test-Logic-Reset state results in synchronous loading of the BYPASS instruction. Asynchronous entry into the Test-Logic-Reset state results in asynchronous loading of the BYPASS instruction. During the Capture-IR TAP controller state, the instruction register is loaded with the value of the previously executed instruction, making this value the register’s read value when the TAP controller is sequenced into the Shift-IR state.

	3	2	1	0
R	Previous Instruction Opcode			
W	Instruction Opcode			
Reset:	BYPASS Instruction Opcode (0xF)			

59.4.1.3 Nexus Device ID (DID) Register

The device identification register, shown in the following figure, allows the part revision number, design center, part identification number, and manufacturer identity code of the part to be determined through the output port.

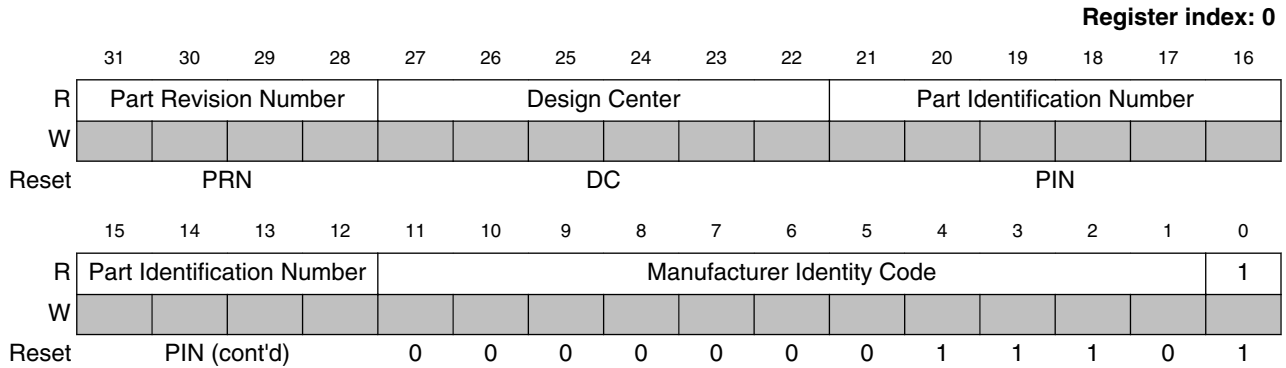


Table 59-7. DID field descriptions

Bit	Name	Description
31:28	PRN	Part Revision Number These bits contain the revision number of the part
27:22	DC	Design Center These bits indicate the device design center
21:12	PIN	Part Identification Number These bits contain the part number of the device
11:1	MIC	Manufacturer Identity Code These bits contain the reduced Joint Electron Device Engineering Council (JEDEC) ID for NXP, 0xE.
0	Bit [0]	IDCODE Register ID Bit [0] identifies this register as the device identification register and not the bypass register

59.4.1.4 Port Configuration Register (PCR)

Register index: 127																
	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R																
W	FPM	MCK O_G T	MCK O_E N	MCKO_DIV			Reserved	DDR_EN	Reserved	0	0	0	Reserved		0	0
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R																
W	Reserved	0	0	0	0	0	Reserved	Reserved	0	0	0	0	0	0	0	PSTA T_EN
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

The PCR, shown in the figure above, is used to select the NPC mode of operation, enable MCKO and select the MCKO frequency, and enable or disable MCKO gating. This register should be configured as soon as the NPC is enabled.

NOTE

The mode or clock division must not be modified after MCKO has been enabled. Changing the mode or clock division while MCKO is enabled can produce unpredictable results.

Table 59-9. PCR field descriptions

Bit	Name	Description
31	FPM	Full Port Mode The value of the FPM bit determines if the auxiliary output port uses the full MDO port or a reduced MDO port to transmit messages. 0 A subset of MDO pins are used to transmit messages 1 All MDO pins are used to transmit messages
30	MCKO_GT	MCKO Clock Gating Control This bit is used to enable or disable MCKO clock gating. If clock gating is enabled, the MCKO clock is gated when the NPC is in enabled mode but not actively transmitting messages on the auxiliary output port. When clock gating is disabled, MCKO is allowed to run even if no auxiliary output port messages are being transmitted. 0 MCKO gating is disabled 1 MCKO gating is enabled
29	MCKO_EN	MCKO Enable This bit enables the MCKO clock to run. When enabled, the frequency of MCKO is determined by the MCKO_DIV field. 0 MCKO clock is driven to zero 1 MCKO clock is enabled

Table continues on the next page...

Table 59-9. PCR field descriptions (continued)

Bit	Name	Description
28:26	MCKO_DIV	MCKO Division Factor The value of this signal determines the frequency of MCKO relative to the system clock frequency when MCKO_EN is asserted. Table 59-10 shows the meaning of MCKO_DIV Values. In this table, SYS_CLK represents the system clock frequency.
25	Reserved	
24	DDR_EN	Double Data Rate Mode Enable This bit enables Nexus double data rate (DDR) mode. In DDR mode, message data is updated between each edge of MCKO, effectively doubling message throughput. NOTE: DDR mode should not be used with Aurora mode 0 DDR mode disabled 1 DDR mode enabled
23	Reserved	
19:18	Reserved	
15	Reserved	
9	Reserved	
8	Reserved	
0	PSTAT_EN	Processor Status Mode Enable¹ This bit enables processor status (PSTAT) mode. In PSTAT mode, all auxiliary output port MDO pins are used to transmit processor status information, and Nexus messaging is unavailable. 0 PSTAT disabled 1 PSTAT mode enabled

1. PSTAT Mode is intended for factory processor debug only. The PSTAT_EN bit should be written to disable PSTAT mode if Nexus messaging is desired. No Nexus messages are transmitted under any circumstances when PSTAT mode is enabled.

Table 59-10. MCKO_DIV Values

MCKO_DIV[2:0]	MCKO Frequency
0	Reserved
1	SYS_CLK/2
2	Reserved
3	SYS_CLK/4
4	Reserved
5	Reserved
6	Reserved
7	SYS_CLK/8

59.5 Functional Description

59.5.1 NPC reset configuration

The NPC is placed in disabled mode upon exit of reset. If message transmission via the output port is desired, a write to the PCR is then required to enable the NPC and select the mode of operation. Asserting MCKO_EN places the NPC in enabled mode and enables MCKO. The frequency of MCKO is selected by writing the MCKO_DIV field. Asserting or negating the FPM bit selects full-port or reduced-port mode, respectively.

The following table describes the NPC reset configuration options.

Table 59-11. NPC reset configuration options

Power On Reset status asserted?	JCOMP asserted high	MCKO_EN bit of the Port Configuration Register	FPM bit of the Port Configuration Register	Configuration
yes	x	X	X	Reset
no	no	X	X	Reset
no	yes	0	X	Disabled
no	yes	1	1	Full-Port Mode
no	yes	1	0	Reduced-Port Mode

59.5.2 Auxiliary Output Port

The auxiliary output port is shared by each of the Nexus modules on the device. The NPC communicates with each of the Nexus modules and arbitrates for access to the port.

59.5.2.1 Output Message Protocol

The protocol for transmitting messages via the auxiliary port is accomplished with the $\overline{\text{MSE0}}$ functions. The $\overline{\text{MSE0}}$ pins are used to signal the end of variable-length packets and the end of messages. They are not required to indicate the end of fixed-length packets. MDO and $\overline{\text{MSE0}}$ are sampled on the rising edge of MCKO.

The following figure illustrates the state diagram for $\overline{\text{MSE0}}$ transfers. All transitions not included in the figure are reserved, and must not be used.

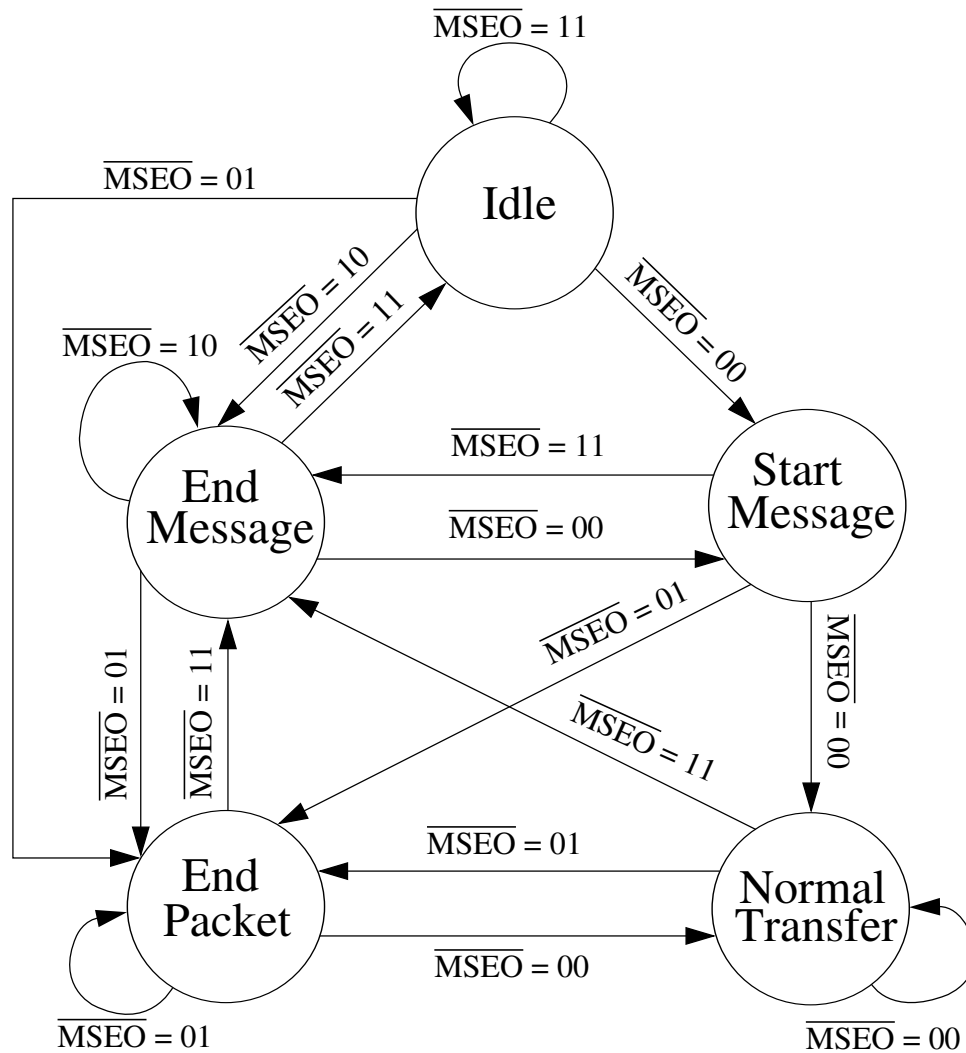


Figure 59-2. MSEO Transfers (for 2-bit MSEO)

59.5.2.2 Output Messages

In addition to sending out messages generated in other Nexus blocks, the NPC can also output the device ID message contained in the device ID register on the MDO pins. The device ID message can also be sent out serially through TDO.

[Figure 59-3](#) describes the device ID output message that the NPC can transmit on the auxiliary port. The TCODE is the first packet transmitted.

Message Name	Min. Packet Size (bits)	Max Packet Size (bits)	Packet Type	Packet Name	Packet Description
Device ID Message	6	6	fixed	TCODE	Value = 1
	32	32	fixed	ID	DID register contents

Figure 59-3. NPC output messages

59.5.2.3 Rules of Message

- A variable-sized field within a message must end on a port boundary. (Port boundaries depend on the number of MDO pins active with the current reset configuration.)
- A variable-sized field may start within a port boundary only when following a fixed-length field.
- Super-fields must end on a port boundary.
- When a variable-length field is sized such that it does not end on a port boundary, it is necessary to extend and zero fill the remaining bits after the highest order bit so that it can end on a port boundary.
- Multiple fixed-length packets may start and/or end on a single clock.
- When any packet follows a variable-length packet, it must start on a port boundary.
- The field containing the TCODE number is always transferred out first, followed by subsequent fields of information.
- Within a field, the lowest significant bits are shifted out first. The following figure shows the transmission sequence of a message that is made up of a TCODE followed by two fields.

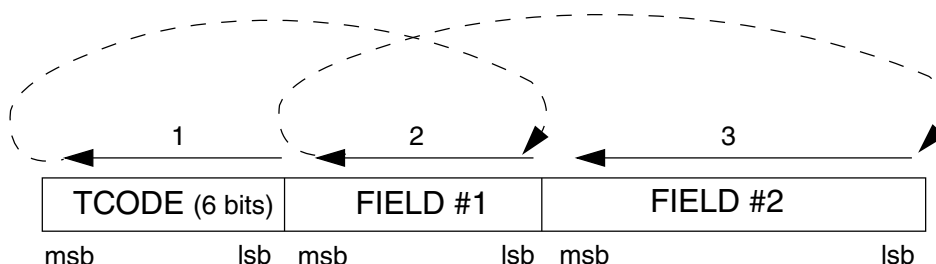


Figure 59-4. Transmission Sequence of Messages

59.5.3 IEEE 1149.1-2001 (JTAG) TAP

The NPC block uses the IEEE 1149.1-2001 test access port (TAP) for accessing registers. Each of the individual Nexus blocks on the device implements a TAP controller for accessing its registers as well. TAP signals include TCK, TDI, TMS, and TDO. There may also be other blocks on the MCU that use the TAP and implement a TAP controller.

Refer to the IEEE 1149.1-2001 specification for further detail on electrical and pin protocol compliance requirements.

The NPC implements a Nexus controller state machine that transitions based on the state of the IEEE 1149.1-2001 state machine shown in [Figure 59-6](#). The Nexus controller state machine is defined by the IEEE-ISTO 5001-2001 standard. It is shown in [Figure 59-7](#).

The instructions implemented by the NPC TAP controller are listed in the following table. The value of the NEXUS-ENABLE instruction is 0b0000. Each unimplemented instruction acts like the BYPASS instruction. The size of the NPC instruction register is 4 bits.

Table 59-12. Implemented Instructions

Instruction Name	Private/Public	Opcode	Description
NEXUS-ENABLE	public	0x0	Activate Nexus controller state machine to read and write NPC registers.
BYPASS	private	0xF	NPC BYPASS instruction. Also the value loaded into the NPC IR upon exit of reset.

Data is shifted between TDI and TDO starting with the least significant bit, as illustrated in the following figure. This applies for the instruction register and all Nexus tool-mapped registers.

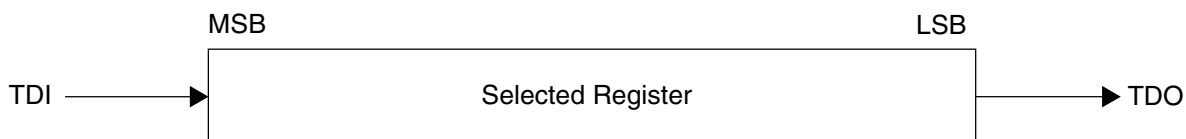
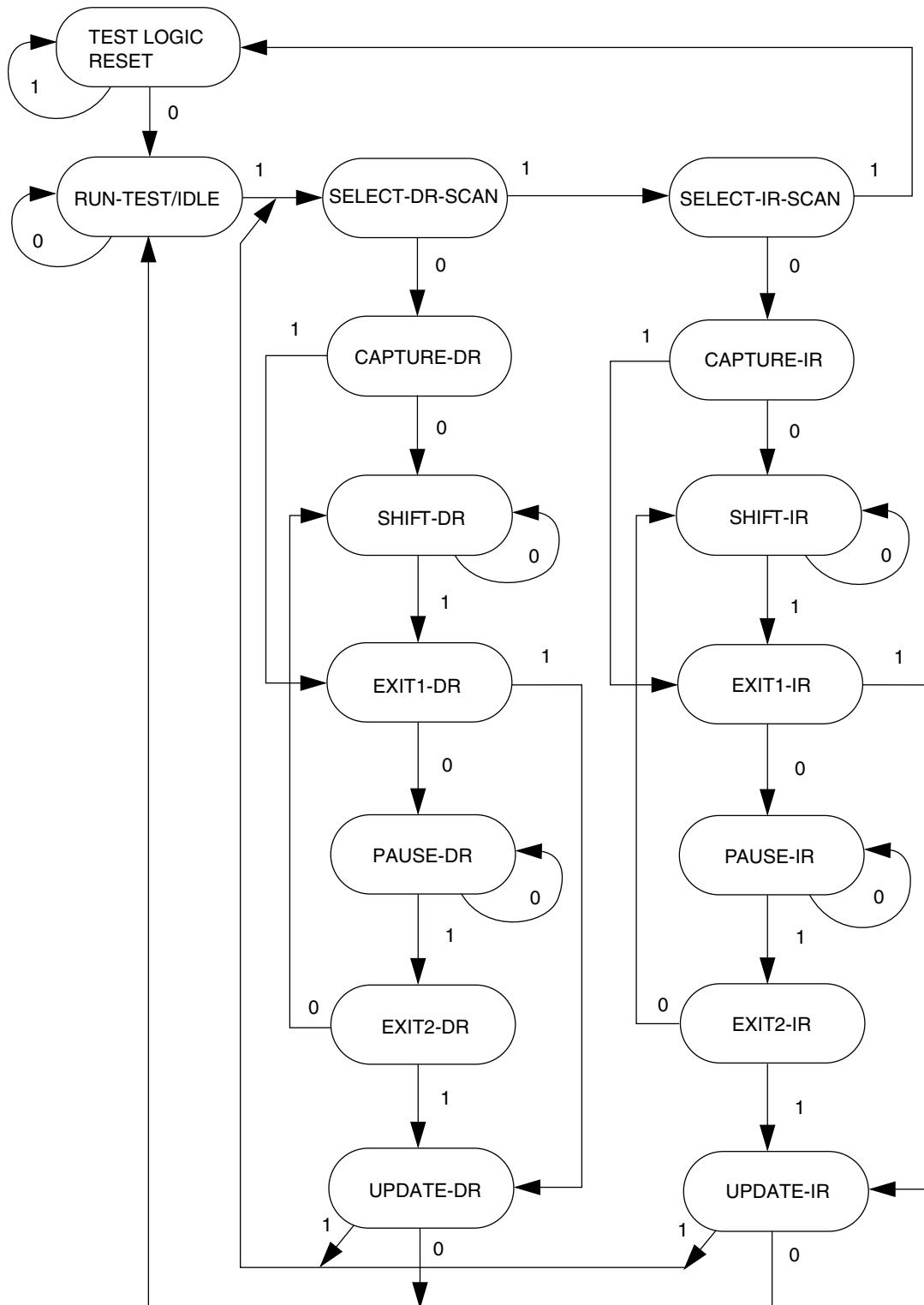


Figure 59-5. Shifting Data Into Register

59.5.3.1 Enabling the NPC TAP controller

Assertion of the power-on reset signal or entry into censored mode or setting JCOMP to a value other than the NPC enable encoding resets the NPC TAP controller. When not in power-on reset or censored mode, the NPC TAP controller is enabled by driving JCOMP with the NPC enable value and exiting the Test-Logic-Reset state. Loading the NEXUS-ENABLE instruction then grants access to Nexus debug.



NOTE: The value shown adjacent to each state transition in this figure represents the value of TMS at the time of a rising edge of TCK.

Figure 59-6. IEEE 1149.1-2001 TAP Controller State Machine

59.5.3.2 Retrieving device IDCODE

The Nexus TAP controller does not implement the IDCODE instruction. However, the device identification message can be output by the NPC through the output port or shifted out serially by accessing the Nexus Device ID register through the TAP.

Transmission of the device identification message on the auxiliary output port MDO pins occurs immediately after a write to the PCR, if the NPC is enabled for auxiliary port mode. Transmission of the device identification message serially via the TAP is achieved by performing a read of the register contents as described in [Selecting a Nexus Client Register](#).

59.5.3.3 Loading NEXUS-ENABLE Instruction

Access to the NPC registers is enabled when the TAP controller instruction register is loaded with the NEXUS-ENABLE instruction. This instruction is shifted in via the SELECT-IR-SCAN path and loaded in the UPDATE-IR state. At this point, the Nexus controller state machine, shown in [Figure 59-7](#), transitions to the REG_SELECT state. The Nexus controller has three states: idle, register select, and data access. [Table 59-13](#) illustrates the IEEE 1149.1 sequence to load the NEXUS-ENABLE instruction.

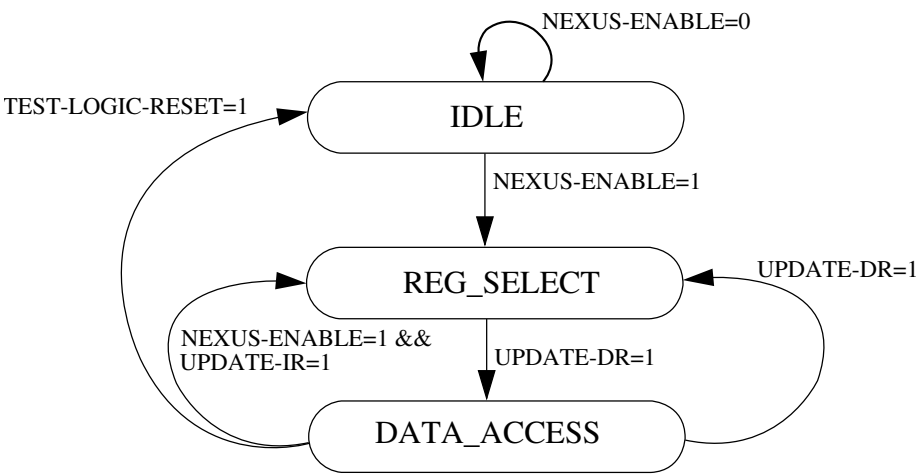


Figure 59-7. NEXUS Controller State Machine

Table 59-13. Loading NEXUS-ENABLE instruction

Clock	TMS	IEEE 1149.1 State	Nexus State	Description
0	0	RUN-TEST/IDLE	IDLE	IEEE 1149.1-2001 TAP controller in idle state
1	1	SELECT-DR-SCAN	IDLE	Transitional state
2	1	SELECT-IR-SCAN	IDLE	Transitional state
3	0	CAPTURE-IR	IDLE	Internal shifter loaded with current instruction

Table continues on the next page...

**Table 59-13. Loading NEXUS-ENABLE instruction
(continued)**

Clock	TMS	IEEE 1149.1 State	Nexus State	Description
4	0	SHIFT-IR	IDLE	TDO becomes active, and the IEEE 1149.1-2001 shifter is ready. Shift in all but the last bit of the NEXUS_ENABLE instruction.
3 TCKS				
8	1	EXIT1-IR	IDLE	Last bit of instruction shifted in
9	1	UPDATE-IR	IDLE	NEXUS-ENABLE loaded into instruction register
10	0	RUN-TEST/IDLE	REG_SELECT	Ready to be read/write Nexus registers

59.5.3.4 Selecting a Nexus Client Register

When the NEXUS-ENABLE instruction is decoded by the TAP controller, the input port allows development tool access to all Nexus registers. Each register has a 7-bit address index.

All register access is performed via the SELECT-DR-SCAN path. The Nexus Controller defaults to the REG_SELECT state when enabled. Accessing a register requires two passes through the SELECT-DR-SCAN path: one pass to select the register and the second pass to read/write the register.

The first pass through the SELECT-DR-SCAN path is used to enter an 8-bit Nexus command consisting of a read/write control bit in the LSB followed by a 7-bit register address index, as illustrated in the following table. The read/write control bit is set to 1 for writes and 0 for reads.

MSB	LSB
7-bit register index	R/W

Figure 59-8. IEEE 1149.1 Controller Command Input

The second pass through the SELECT-DR-SCAN path is used to read or write the register data by shifting in the data (LSB first) during the SHIFT-DR state. When reading a register, the register value is loaded into the JTAG shifter during the CAPTURE-DR state. When writing a register, the value is loaded from the IEEE 1149.1-2001 shifter to the register during the UPDATE-DR state. When reading a register, there is no requirement to shift out the entire register contents. Shifting may be terminated once the required number of bits have been acquired.

The following table illustrates a sequence which writes a 32-bit value to a register.

Table 59-14. Write to a 32-Bit Nexus Client Register

Clock	TMS	IEEE 1149.1 State	Nexus State	Description
0	0	RUN-TEST/IDLE	REG_SELECT	IEEE 1149.1-2001 TAP controller in idle state
1	1	SELECT-DR-SCAN	REG_SELECT	First pass through SELECT-DR-SCAN path
2	0	CAPTURE-DR	REG_SELECT	Internal shifter loaded with current value of controller command input.
3	0	SHIFT-DR	REG_SELECT	TDO becomes active, and write bit and 6 bits of register index shifted in.
7 TCKs				
11	1	EXIT1-DR	REG_SELECT	Last bit of register index shifted into TDI
12	1	UPDATE-DR	REG_SELECT	Controller decodes and selects register
13	1	SELECT-DR-SCAN	DATA_ACCESS	Second pass through SELECT-DR-SCAN path
14	0	CAPTURE-DR	DATA_ACCESS	Internal shifter loaded with current value of register
15	0	SHIFT-DR	DATA_ACCESS	TDO becomes active, and outputs current value of register while new value is shifted in through TDI
31 TCKs				
47	1	EXIT1-DR	DATA_ACCESS	Last bit of current value shifted out TDO. Last bit of new value shifted in TDI.
48	1	UPDATE-DR	DATA_ACCESS	Value written to register
49	0	RUN-TEST/IDLE	REG_SELECT	Controller returned to idle state. It could also return to SELECT-DR-SCAN to write another register.

59.5.4 Nexus JTAG Port Sharing

Each of the individual Nexus blocks on the device implements a TAP controller for accessing its registers. When Nexus has ownership of the TAP, only the block whose NEXUS-ENABLE instruction is loaded has control of the TAP. This allows the interface to all of these individual TAP controllers to appear to be a single port from outside the device. If no register is selected as the shift path for a Nexus block, that block acts like a single-bit shift register, or bypass register.

59.5.5 MCKO Control

MCKO is an output clock to the development tools used for the timing of $\overline{\text{MSEO}}$ and MDO pin functions. MCKO is derived from the system clock and its frequency is determined by the value of the MCKO_DIV field in the PCR. Possible operating frequencies include system clock, one-half system clock, one-quarter system clock, and one-eighth system clock speed.

The NPC also generates an MCKO clock gating control output signal. This output can be used by the MCKO generation logic to gate the transmission of MCKO when the auxiliary port is enabled but not transmitting messages. The setting of the MCKO_GT bit inside the PCR determines whether or not MCKO gating control is active. The MCKO_GT bit resets to a logic 0. In this state gating of MCKO is disabled. To enable gating of MCKO, the MCKO_GT bit in the PCR is written to a logic 1.

59.5.6 $\overline{\text{EVTO}}$ Sharing

The NPC block controls sharing of the $\overline{\text{EVTO}}$ output between all Nexus clients that produce an $\overline{\text{EVTO}}$ signal. The NPC assumes incoming $\overline{\text{EVTO}}$ signals will be asserted for one system clock period. After receiving a single clock period of asserted $\overline{\text{EVTO}}$ from any Nexus client, the NPC latches the result, and drives $\overline{\text{EVTO}}$ for one MCKO period on the following clock. When there is no active MCKO, such as in disabled mode, the NPC drives $\overline{\text{EVTO}}$ for two system clock periods. $\overline{\text{EVTO}}$ sharing is active as long as the NPC is not in reset.

59.5.7 Nexus Reset Control

The JCOMP input that is used as the primary reset signal for the NPC is also used by the NPC to generate a single-bit reset signal for other Nexus blocks. If JCOMP is negated, an internal reset is asserted, indicating that all Nexus modules should be held in reset. Internal Nexus reset is also asserted when the device is in censored mode.

59.5.8 System Clock Locked Indication

Following a power-on reset, MDO[0] can be monitored to provide the lock status of the system clock. MDO[0] is driven to a logic 1 until the system clock achieves lock after exiting power-on reset. Once the system clock is locked, MDO[0] is negated and tools may begin Nexus configuration. Loss of lock conditions that occur subsequent to the exit of power-on reset and the initial lock of the system clock do not cause a Nexus reset, and therefore do not result in MDO[0] driven high.

59.6 Initialization/Application Information

59.6.1 Accessing NPC tool-mapped registers

To initialize the TAP for Nexus register accesses, the following sequence is required:

1. Enable the Nexus TAP controller
2. Load the TAP controller with the NEXUS-ENABLE instruction

To write control data to NPC tool-mapped registers, the following sequence is required:

1. Write the 7-bit register index and set the write bit to select the register with a pass through the SELECT-DR-SCAN path in the TAP controller state machine.
2. Write the register value with a second pass through the SELECT-DR-SCAN path. Note that the prior value of this register is shifted out during the write.

To read status and control data from NPC tool-mapped registers, the following sequence is required:

1. Write the 7-bit register index and clear the write bit to select register with a pass through SELECT-DR-SCAN path in the TAP controller state machine.
2. Read the register value with a second pass through the SELECT-DR-SCAN path. Data shifted in is ignored.

See the IEEE-ISTO 5001-2001 standard for more detail.

Chapter 60

Nexus eTPU Development Interface (NDEDI)

60.1 Chip-specific NDEDI information

This chip contains two Nexus eTPU Development Interface modules.

- A dual NDEDI covers eTPU_A and eTPU_B. The NDEDI's ENGINE1 corresponds to eTPU_A and ENGINE2 corresponds to eTPU_B.
- A single Nexus eTPU Development Interface (NSEDI) covers eTPU_C. The single interface has the same features as the NDEDI but is for a single eTPU engine.

Each interface also includes a coherent dual-parameter controller (CDC) for coherent access to eTPU parameters. In each interface, the eTPU modules and CDC unit appear as separate Nexus clients.

60.1.1 NDEDI/NSEDI device-specific parameters

The following table provides the chip-specific parameter values of the NDEDI module and NSEDI module. Each parameter has a placeholder "variable" name that appears in the detailed NDEDI description.

Table 60-1. Device-specific parameters

Parameter	Variable name	Description	NDEDI value	NSEDI value
Full-Port Mode MDO pins	N	Number of MDO pins used for message transmission in full-port mode: This value is the same for all Nexus blocks sharing the port.	16d	16d
Reduced-Port Mode MDO pins	J	Number of MDO pins used for message transmission in reduced-port mode: This value is the same for all Nexus blocks sharing the port.	12d	12d
SRC Packet size	K	Number of bits used for the SRC packet in messages	4d	4d

Table continues on the next page...

Table 60-1. Device-specific parameters (continued)

Parameter	Variable name	Description	NDEDI value	NSEDI value
HIST Packet size	H	Number of bits used for the HIST packet in messages	26d	26d
Queue size	Q_SIZE	Depth of the event queue used to store events before transmission	32d	32d
Stall Threshold	STALL_THRESHOLD	Number of spots within the queue used as a threshold to delay the processor	7d	7d
ENGINE1 SRC value	ENGINE1_SRC	SRC packet value for messages generated by ENGINE1	2h	Ah
ENGINE2 SRC value	ENGINE2_SRC	SRC packet value for messages generated by ENGINE2	3h	—
CDC SRC value	CDC_SRC	SRC packet value for messages generated by the CDC	4h	Ch
Manufacturer ID	DID_MIC	Manufacturer code used in the DID register	Eh	Eh
Design Center ID	DID_DC	Design center code used in the DID register	20h	20h
Part Revision Number	DID_PRN	Part revision number used in the DID register	4h	4h
Part Identification Number	DID_PIN	Part identification number used in the DID register	224h	124h

60.2 Introduction

The Nexus ETPU Development Interface (NDEDI) provides real-time development capabilities for the ETPU system including two engines and the coherent dual-parameter controller (CDC) in compliance with the IEEE-ISTO 5001-2002 standard. The NDEDI interfaces to the ETPU via dedicated trace busses. The main development features supported are breakpoint/watchpoint configuration, debug mode register access, enter debug mode at reset negation or during normal execution, single stepping of instructions, branch trace, data trace, ownership trace, and watchpoint trace. Combined, these features make the interface for each engine compliant with Class 3 of the IEEE-ISTO 5001-2002 standard.

Both engines have their own Nexus register sets that allows trace to be set up independently for each of them. The only exception to this is the data trace address range registers that are shared. All the registers are accessed via the JTAG port. Output messages between the engines and other sources are differentiated by the value of the SRC packet in the output messages.

The NDEDI block is combined with similar blocks for other intelligent peripherals, a read/write access block, and a Nexus Port Controller block to provide the complete IEEE-ISTO 5001-2002 interface for a system-on-a-chip design. The JTAG port and Nexus auxiliary port pins are shared by all of the Nexus based blocks on the chip.

60.3 Features

The NDEDI block is compliant with the IEEE-ISTO 5001-2002 standard, and implements the following features:

- Full duplex pin interface for medium and high visibility throughput
 - Pin interface shared among all Nexus-based development interfaces on the device
 - One of two modes selected during reset: full-port mode (FPM) and reduced-port mode (RPM)
- Auxiliary Output Port
 - 1 MCKO (Message Clock Out) pin
 - MDO (Message Data Out) port (port size is SoC specific)
 - 2 $\overline{\text{MSEO}}$ (Message Start/End Out) pins
 - 1 $\overline{\text{EVTO}}$ (Event Out) pin
- Auxiliary Input Port
 - 1 $\overline{\text{EVTI}}$ (Event In) pin
- IEEE 1149.1 (JTAG) Test Access Port (TAP)
 - Support for optional Multi-JTAG TAP Linking Module (TLM)
 - 4 pins (TDI, TDO, TMS, and TCK)
 - Reset input $\overline{\text{TRST}}$ driven by either the Nexus Port Controller or an external pin
- ETPU Development Support
 - IEEE-ISTO 5001-2002 standard class 3 compliant for both engines
 - Read/Write registers in debug mode
 - Ability to enter debug mode at reset negation or during normal execution

- Ability to single step an instruction and re-enter debug mode
- Breakpoint and Watchpoint Configuration
- Execute external microcode instructions in debug mode providing indirect read/write accesses to ETPU registers
- Read the microprogram counter in debug mode
- Change the microprogram counter by executing external microcode instructions in debug mode
- Data Trace via Data Write Messaging (DWM) and Data Read Messaging (DRM). This allows the development tool to trace reads and writes to selected shared parameter RAM (SPRAM) address ranges
 - Four data trace windows with programmable address ranges and access attributes are provided. Data trace windowing reduces the requirement on the auxiliary port bandwidth by constraining the number of trace locations. The four trace window address ranges are shared among the dual engines and the ETPU Coherent Dual-Parameter Controller (CDC)
- Ownership Trace via Ownership Trace Messaging (OTM). OTM provides visibility of which channel is being serviced. An ownership trace message is transmitted to indicate when a new channel service request is scheduled, allowing the development tools to trace task flow. A special OTM is sent when the engine enters in idle, meaning that all requests were serviced and no new requests are yet scheduled
- Program Trace via Branch Trace Messaging (BTM). BTM displays program flow discontinuities (start, jump, return, etc....), allowing the development tool to interpolate what transpires between the discontinuities. Thus static code may be traced
 - Public messages are used for jump, and return instructions
 - Private messages are used to indicate special ETPU cases not covered by public messages. These messages include the writes to the channel register, the start of a channel service and the trace enabling after a watchpoint hit
 - Branch/Predicate History is kept on direct branches (taken and not taken) and predicated instructions (executed or not executed) to minimize the number of messages transmitted
- Watchpoint Messaging (WPM) via the auxiliary port. WPM provides visibility of the occurrence of the ETPUs' watchpoints

- Event Queue Interface. The Event Queue array is located external to the NDEDI block to separate hard and soft IP. The depth of the Event Queue is specified in the SoC Guide
- Read/Write Access functions are provided on the device level by a block separate from the NDEDI block

60.4 Modes of operation

60.4.1 Reset

The NDEDI block is placed in reset when Nexus is reset, or the TAP controller state machine is in the Test-Logic-Reset state. Holding TMS high for 5 consecutive rising edges of TCK guarantees entry into the Test-Logic-Reset state regardless of the current TAP controller state. Nexus reset results in asynchronous entry into the reset state. The NDEDI is unaffected by other sources of reset. While in reset, the following actions occur:

- The TAP controller is forced into the Test-Logic-Reset state
- The auxiliary output port pins are negated
- The auxiliary output port enable outputs are negated
- The TDI, TMS, and TCK TAP inputs are ignored
- Registers default back to their reset values

Note that the NDEDI is not reset by the system reset signal. The system reset signal is used though to make ETPU entering in debug mode from reset condition and to synchronize Program Trace and Data Trace messages.

60.4.2 Disabled-port mode

In disabled-port mode, auxiliary output pin port enable signals are negated, thereby disabling message transmission. Thus, the NDEDI class 3 features are not available in this mode. The primary features available are class 1 features and read/write access to the registers.

60.4.3 Full-port mode

Full-port mode (FPM) is determined by the chip Nexus configuration. See the chip-specific NPC information. In full-port mode, the block is enabled and the FPM port enable is asserted indicating that all available MDO pins are to be used for message transmission. All trace features are enabled or can be enabled by writing the configuration registers via the TAP. The number of MDO pins available is device-specific. Refer to the SoC guide for this information.

60.4.4 Reduced-port mode

Reduced-port mode (RPM) is determined by the chip Nexus configuration. See the chip-specific NPC information. In reduced-port mode, the block is enabled and the RPM port enable is asserted indicating that only a subset of the MDO pins are to be used for message transmission. All trace features are enabled or can be enabled by writing the configuration registers via the TAP. The number of MDO pins available is device-specific. Refer to the SoC guide for this information.

60.4.5 Stop mode

The NDEDI supports the IPI Green-Line Interface stop mode mechanism. When a request is made to enter stop mode, the NDEDI block waits for the ETPU to acknowledge the stop condition so no trace information is lost. The NDEDI then completes transmitting all pending messages, emptying the Event Queue. Finally, NDEDI transmits a debug status message to indicate it is entering stop mode, and if Program Trace is enabled it will send a correlation message. Refer to [Functional description](#) for further information about the messages. After this is completed, the request to enter stop mode is acknowledged. No breakpoint requests are sent to the engines while they are in stop mode. After the acknowledgement, the system clock input may be shut off by the clock driver on the device. While the clocks are shut off, the development tool must not access NDEDI registers via the JTAG test access port.

60.5 External signal description

The following section describes the External Signal description.

60.5.1 Overview

The NDEDI pin provides interface for the transmission of messages and for accessing Nexus registers. The NDEDI pin definition is outlined in the following table.

Table 60-2. NDEDI signal properties

Name	Port	Function	Reset State	Pull ¹
EVTI	Auxiliary	Event In pin	—	Up
EVTO	Auxiliary	Event Out pin	1	—
MCKO	Auxiliary	Message Clock Out pin (from NPC)	—	—
MDO[J-1:0] or MDO[N-1:0]	Auxiliary	Message Data Out pins	0	—
MSEO[1:0]	Auxiliary	Message Start/End Out pins	0b11	—
TCK	TAP	Test Clock Input	—	—
TDI	TAP	Test Data Input	—	Up
TDO	TAP	Test Data Output	0	—
TMS	TAP	Test Mode Select Input	—	Up
TRST	TAP	Test Reset Input	—	Up

1. The pull is not implemented in this block. Refer to the SoC guide for more details.

60.5.2 Detailed signal descriptions

This section describes each of the signals listed in the preceding table in more detail.

60.5.2.1 $\overline{\text{EVTI}}$ - Event In

Event In ($\overline{\text{EVTI}}$) is used to initiate program and data trace synchronization messages or to generate a breakpoint. $\overline{\text{EVTI}}$ is edge-sensitive for synchronization and breakpoint generation.

60.5.2.2 $\overline{\text{EVTO}}$ - Event Out

Event Out ($\overline{\text{EVTO}}$) is an output pin to development tools comprising exact timing for a single watchpoint or breakpoint occurrence. $\overline{\text{EVTO}}$ is asserted for 1 MCKO clock¹.

1. NDEDI asserts The EVTO indication for 1 system clock.

60.5.2.3 MCKO - Message Clock Out

Message Clock Out (MCKO) is an output clock to the development tools used for the timing of MDO and $\overline{\text{MSEO}}$ pin functions. The frequency of MCKO is determined by the NPC block. Refer to the NPC BG for more information.

The MCKO signal is generated external to the NDEDI block but is included in this document for functional clarity. An input is provided to this block to keep logic synchronized to MCKO.

60.5.2.4 MDO - Message Data Out

Message Data Out (MDO) are output pins used for uploading messages to the development tool. The development tool must sample MDO on the rising edge of MCKO. The width of the MDO bus is determined by the Nexus full port mode configuration. See the chip-specific NPC information for further information.

60.5.2.5 MSEO[1:0] - Message Start/End Out

Message Start/End Out ($\overline{\text{MSEO}}$) is an output pin that indicates when a message on the MDO pins has started, when a variable length packet has ended, or when the message has ended. The development tool must latch $\overline{\text{MSEO}}$ on the rising edge of MCKO.

60.5.2.6 TCK - Test Clock Input

Test Clock Input (TCK) is an input pin used to synchronize the test logic and control register access through the JTAG port.

60.5.2.7 TDI - Test Data Input

Test Data Input (TDI) is an input pin that receives serial test instructions and data. TDI is sampled on the rising edge of TCK.

60.5.2.8 TDO - Test Data Output

Test Data Output (TDO) is an output pin that transmits serial output for test instructions and data. TDO is actively driven in the 'SHIFT-IR' and 'SHIFT-DR' states of the JTAG state machine. TDO changes on the falling edge of TCK and is sampled on the rising edge of TCK.

60.5.2.9 TMS - Test Mode Select

Test Mode Select (TMS) is an input pin used to sequence the JTAG state machine. TMS is sampled on the rising edge of TCK.

60.5.2.10 $\overline{\text{TRST}}$ - Test Reset Input

The Test Reset Input is used to asynchronously reset the NDEDI block and initialize the JTAG controller. This signal can be generated in the Nexus Port Controller or driven by an external pin. Refer to the SoC Guide for more information.

60.6 Register definition

This section provides a detailed description of all NDEDI registers accessible to the development tool. Individual bit-level descriptions and reset states of each register are included.

The following table shows the NDEDI registers by client select and index values. These registers are not memory-mapped and can only be accessed via the TAP.

Table 60-3. NDEDI Registers

CS	Index	Register
Client-Independent Registers		
X	0	Device Identity (DID)
X	1	Client Select Control (CSC)
ENGINE1 Control/Status Registers		
ENGINE1_SRC	2	ENGINE1 Development Control (NDEDI_ENGINE1_DC)
ENGINE1_SRC	4	ENGINE1 Development Status (NDEDI_ENGINE1_DS)
ENGINE1_SRC	11	ENGINE1 Watchpoint Trigger (NDEDI_ENGINE1_WT)
ENGINE1_SRC	13	ENGINE1 Data Trace Control (NDEDI_ENGINE1_DTC)
ENGINE1_SRC	22	ENGINE1 Breakpoint/Watchpoint Control 1 (NDEDI_ENGINE1_BWC1)
ENGINE1_SRC	23	ENGINE1 Breakpoint/Watchpoint Control 2 (NDEDI_ENGINE1_BWC2)

Table continues on the next page...

Table 60-3. NDEDI Registers (continued)

CS	Index	Register
ENGINE1_SRC	24	ENGINE1 Breakpoint/Watchpoint Control 3 (NDEDI_ENGINE1_BWC3)
ENGINE1_SRC	30	ENGINE1 Breakpoint/Watchpoint Address 1 (NDEDI_ENGINE1_BWA1)
ENGINE1_SRC	31	ENGINE1 Breakpoint/Watchpoint Address 2 (NDEDI_ENGINE1_BWA2)
ENGINE1_SRC	38	ENGINE1 Breakpoint/Watchpoint Data 1 (NDEDI_ENGINE1_BWD1)
ENGINE1_SRC	39	ENGINE1 Breakpoint/Watchpoint Data 2 (NDEDI_ENGINE1_BWD2)
ENGINE1_SRC	64	ENGINE1 Program Trace Channel Enable (NDEDI_ENGINE1_PTCE)
ENGINE1_SRC	69	ENGINE1 Microinstruction Debug Register (NDEDI_ENGINE1_INST)
ENGINE1_SRC	70	ENGINE1 Microinstruction Program Counter (NDEDI_ENGINE1_MPC)
ENGINE1_SRC	71	ENGINE1 Channel Flag Status Register(NDEDI_ENGINE2_CSFR)
ENGINE2 Control/Status Registers		
ENGINE2_SRC	2	ENGINE2 Development Control (NDEDI_ENGINE2_DC)
ENGINE2_SRC	4	ENGINE2 Development Status (NDEDI_ENGINE2_DS)
ENGINE2_SRC	11	ENGINE2 Watchpoint Trigger (NDEDI_ENGINE2_WT)
ENGINE2_SRC	13	ENGINE2 Data Trace Control (NDEDI_ENGINE2_DTC)
ENGINE2_SRC	22	ENGINE2 Breakpoint/Watchpoint Control 1 (NDEDI_ENGINE2_BWC1)
ENGINE2_SRC	23	ENGINE2 Breakpoint/Watchpoint Control 2 (NDEDI_ENGINE2_BWC2)
ENGINE2_SRC	24	ENGINE2 Breakpoint/Watchpoint Control 3 (NDEDI_ENGINE2_BWC3)
ENGINE2_SRC	30	ENGINE2 Breakpoint/Watchpoint Address 1 (NDEDI_ENGINE2_BWA1)
ENGINE2_SRC	31	ENGINE2 Breakpoint/Watchpoint Address 2 (NDEDI_ENGINE2_BWA2)
ENGINE2_SRC	38	ENGINE2 Breakpoint/Watchpoint Data 1 (NDEDI_ENGINE2_BWD1)
ENGINE2_SRC	39	ENGINE2 Breakpoint/Watchpoint Data 2 (NDEDI_ENGINE2_BWD2)
ENGINE2_SRC	64	ENGINE2 Program Trace Channel Enable (NDEDI_ENGINE2_PTCE)
ENGINE2_SRC	69	ENGINE2 Microinstruction Debug Register (NDEDI_ENGINE2_INST)
ENGINE2_SRC	70	ENGINE2 Microinstruction Program Counter (NDEDI_ENGINE2_MPC)
ENGINE2_SRC	71	ENGINE2 Channel Flag Status Register(NDEDI_ENGINE2_CSFR)
ETPU CDC Control/Status Registers		
CDC_SRC	13	CDC Data Trace Control (NDEDI_CDC_DTC)
ENGINE1/ENGINE2/CDC Shared Control/Status Registers		
ENGINE1_SRC or ENGINE2_SRC or CDC_SRC	65	NDEDI Data Trace Address Range 0 (NDEDI_DTAR0)
ENGINE1_SRC or ENGINE2_SRC or CDC_SRC	66	NDEDI Data Trace Address Range 1 (NDEDI_DTAR1)
ENGINE1_SRC or ENGINE2_SRC or CDC_SRC	67	NDEDI Data Trace Address Range 2 (NDEDI_DTAR2)
ENGINE1_SRC or ENGINE2_SRC or CDC_SRC	68	NDEDI Data Trace Address Range 3 (NDEDI_DTAR3)

60.6.1 Register descriptions

This section consists of NDEDI register descriptions. The ENGINE1 and ENGINE2 registers are shown in the preceding table. To simplify the document, the description of registers with the same content is only given once instead of being repeated for each ENGINE. This is denoted in the text as ENGINE_x where ENGINE_x can be replaced by either ENGINE1 or ENGINE2.

60.6.1.1 Device Identity Register (DID)

The DID register contains the unique identification information.

Client Select: X

Register index: 0

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	Part Revision Number				Design Center						Part Identification Number					
W																
RESET:	DID_PRN				DID_DC						DID_PIN					
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	Part Identification Number				Manufacturer Identity Code											
W																
RESET:	DID_PIN (contd)				0	0	0	0	0	0	0	0	1	1	1	0
																1

Figure 60-1. Device Identity Register (DID)

Table 60-4. DID field descriptions

Field	Description
PRN	Part Revision Number Bits[31:28] contain the revision number of the part.
DC	Design Center Bits [27:22] indicate the Freescale Design Center.
PIN	Part Indication Number Bits[21:12] contain the part number of the device.
MIC	Manufacturer Identity Code Bits [11:1] contain the reduced Joint Electron Device Engineering Council (JEDEC) ID for Freescale.
Bit[0]	IDCODE Register ID Bit [0] identifies this register as the device identification register and not the bypass register.

60.6.1.2 Client Select Control Register (CSC)

Client Select: X

Register index: 1

Register definition

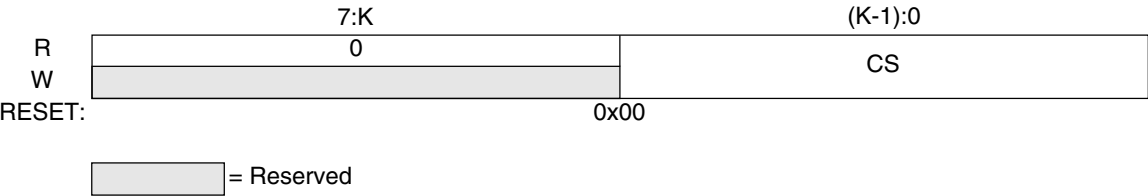


Figure 60-2. Client Select Control Register (CSC)

Table 60-5. CSC field decription

Field	Description
CS	Client Select The value of the CS field determines which client is accessed. The number of bits implemented for the CS field (K) is MCU dependent, and is the same number of bits as the size of the SRC field transmitted with messages. Refer to the SoC Guide for the size on a particular MCU. The value written to the CSC register to access a client register is the same value as the SRC field a client uses in its messages. For example, to access an NDEDI ENGINE1 register, the CSC register is written with a value of ENGINE1_SRC.

60.6.1.3 ENGINE_x Development Control Register (NDEDI_ENGINE_x_DC)

The NDEDI_ENGINE_x_DC register controls various trace and debug features as described below.

When a field is set to a reserved value the behavior defaults to the reset values behavior.

Client Select: ENGINE_x_SRC

Register index: 2

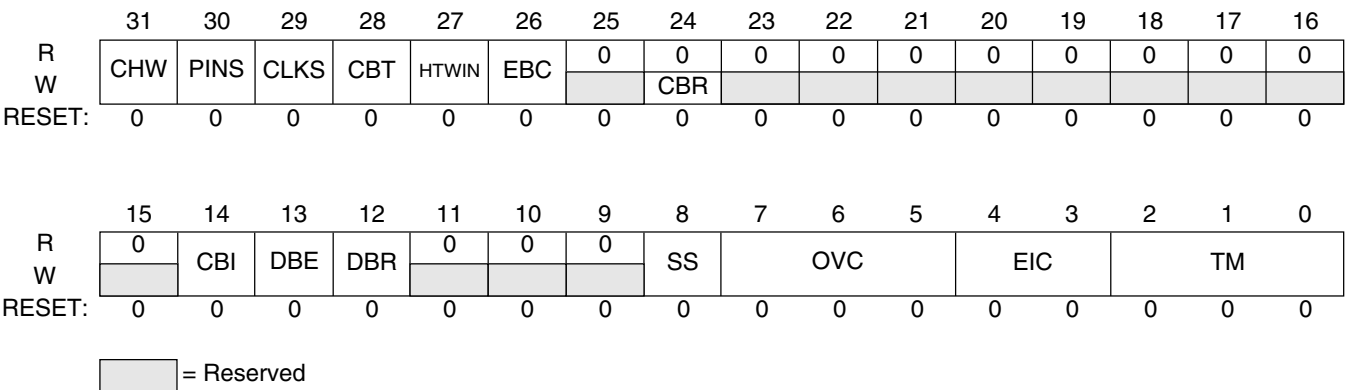


Figure 60-3. ENGINE_x Development Control Register (NDEDI_ENGINE_x_DC)

Table 60-6. NDEDI_ENGINEx_DC field descriptions

Field	Description
CHW	ENGINEx CHAN Register Write Trace The CHW bit enables the tracing of writes to the CHAN register in the ENGINEx execution unit. CHAN register write tracing requires the channel being serviced to have program trace enabled. 0 = Tracing not enabled for ENGINEx CHAN writes. 1 = Tracing enabled for ENGINEx CHAN writes
PINS	Stop Pins in Debug Mode The PINS bit controls whether the ENGINEx pins are sampled when the ENGINEx enters debug mode. When PINS is set, the pins are not sampled during debug mode or when executing a forced instruction from the microinstruction register. The pins are sampled during single steps. 0 = Sample pins in the halted state 1 = Stop sampling pins in the halted state
CLKS	Stop TCR Clocks The CLKS bit controls whether the TCR clocks from ENGINEx stop running when the ENGINEx enters debug mode. When CLKS is set, TCR is not updated during debug mode or when executing a forced instruction from the microinstruction register. TCR is updated during single steps. 0 = Do not stop TCRs during debug mode 1 = Stop TCRs during debug mode
CBT	Client Breakpoint Timing The CBT bit controls the timing of an ENGINEx halt due to an external breakpoint source or due to a breakpoint at the twin engine. For this bit to matter, CBI or HTWIN must be set to 1. 0 = Halt at the end of the current microcycle 1 = Halt at the completion of the current instruction thread
HTWIN	HTWIN — Halt on Twin Engine Breakpoint Along with CBT, the HTWIN bit controls the action taken on breakpoints occurrences at the twin engine. If the Twin Engine that is causing a breakpoint exit debug state, the ENGINEx shall resume its operation. If the HTWIN bit is cleared while the ENGINEx is in debug state due to a Twin Engine Breakpoint, the ENGINEx shall resume its operation. This bit must not be cleared when the engine is neither halted nor stopped. 0 = Do not halt for breakpoints in the twin Engine 1 = Halt for twin Engine's breakpoint
EBC	EVT$\overline{O}$ Breakpoint Controller The EBC bit controls the generation of EVT$\overline{O}$ due to any of the following breakpoint types: Exit of System Reset, DBR assertion, SW, TWIN Engine, EVTI, MISC Error, Illegal Instruction, Single Step or External. 0 = Breakpoint status indication is not output on EVT$\overline{O}$ 1 = Breakpoint status indication is output on EVT$\overline{O}$

Table continues on the next page...

Table 60-6. NDEDI_ENGINEx_DC field descriptions (continued)

Field	Description
CBR	Clear Breakpoint Request Writing this bit to one clears breakpoint requests caused by the following conditions: • Exit of system reset • Internal breakpoint (due to: instruction address match, accesses to the SPRAM, channel service and writes to the CHAN register) • Single Step breakpoint • EVTI assertion • Halt instruction at the engine • MISC miscompare • Illegal instruction at the engine. Asserting this bit causes it to clear one clock later. This means that the development tool can never read this bit with a value other than zero. The ETPU must always execute the first instruction after exiting a breakpoint condition, in such a way that it does not halt twice at the same position. The breakpoints caused by external/system debug request or twin engine are not cleared by the assertion of this bit. This bit must not be asserted when the engine is not halted. 0 = No action 1 = Clear breakpoint requests
CBI	Client Breakpoint Input Along with CBT, the CBI bit controls the action taken on breakpoints from external sources. The breakpoint condition caused by the external source is only cleared if CBI equals 0 or if the external breakpoint is negated. This bit must not be cleared when the engine is neither halted nor stopped. 0 = Do not halt for other clients' breakpoints 1 = Halt for other clients' breakpoints

Table continues on the next page...

Table 60-6. NDEDI_ENGINEx_DC field descriptions (continued)

Field	Description
DBE	Debug Enable DBE enables debug mode. When DBE is asserted, the occurrence of any of the following events will cause the processor to enter debug mode: • DBR assertion • Twin Engine breakpoint • External breakpoint • Exit of system reset • Internal breakpoint (due to: instruction address match, accesses to the SPRAM, channel service and writes to the CHAN register) • Single Step breakpoint • EVTI assertion • Halt instruction at the engine • MISC miscompare • Illegal instruction at the engine. No breakpoints will occur if DBE is not asserted. If this bit is negated while in debug mode a request to resume the processor operation will be sent to the engine. This bit must not be cleared when the engine is either halted nor stopped. This bit is necessary to use features such as single stepping and breakpoints, but does not affect neither watchpoint, program trace nor data trace operations. 0 = Debug mode disabled 1 = Debug mode enabled
DBR	Debug Request Mechanism to allow development tool to request processor to enter debug mode. When written, and DBE equals 1 this bit forces the ENGINEx to enter debug mode. The engine will not exit debug mode while this bit is asserted and DBE equals 1. 0 = Exit debug mode if there are no pending breakpoint sources 1 = Request debug mode

Table continues on the next page...

Table 60-6. NDEDI_ENGINEx_DC field descriptions (continued)

Field	Description
OVC	OVC — Overrun Control The NDEDI can be programmed to stall the ENGINEx operation avoiding errors related to that engine due to full Event Queue. The ENGINEx can't be stalled neither in the middle of an atomic sequence of accesses to the SPRAM nor in the middle of the Time Slot Transition(TST). Please refer to the ETPU BG for further information. The overrun control field controls how the NDEDI reacts when the Queue is about to get full. If programmed to delay the processor, the NDEDI asserts the request to delay the processor when the Event Queue has only STALL_THRESHOLD available positions, and negates the delay request as soon as there are STALL_THRESHOLD + 1 available positions. The request to delay the processor does not affect neither the other engine nor the CDC operations. Therefore, in some extreme cases an overrun message may be generated even with OVC configured to delay the processor. If an error event happens, the processor will be delayed until the Event Queue is enabled for storing snapshots again, which will happen as the Event Queue gets empty. If the OVC field is changed to 0b011 while the Event Queue has less than STALL_THRESHOLD available positions the NDEDI will automatically assert the stall request. If the OVC field is changed to a value other than 0b011 while the ENGINEx is in STALL state, the NDEDI will automatically negate the stall request. A breakpoint request prevails over the request to delay the processor. Thus, if a breakpoint request happens at the same time of a stall request, the ETPU will enter in HALT state instead of STALL state. OVC values: 000b = Generate overrun message 001b–010b = Reserved 011b = Stall Processor to prevent overruns for ENGINEx 100b–111b = Reserved
EIC	ENGINEx $\overline{\text{EVTI}}$ Control The $\overline{\text{EVTI}}$ control (EIC) field can be configured for synchronization or breakpoint generation. The $\overline{\text{EVTI}}$ signal is edge sensitive. ENGINEx_EIC values: 00b = $\overline{\text{EVTI}}$ assertion causes the next ENGINEx program and data trace message to be a synchronization message (the next CDC data trace messages are synchronized if the next message from either Engines are synchronized). 01b = $\overline{\text{EVTI}}$ assertion causes ENGINEx breakpoint generation 10b = No operation 11b = Reserved
TM	Trace Mode The TM field enables BTM, DTM, and OTM for ENGINEx. One or all types of trace may be enabled at a single time by writing to the TM field. Note: Data and Program Trace can also be enabled through watchpoint triggers. The two enable sources are ORed together to form the true trace enable. TM values: 000b = No trace enabled 1xxb = BTM enabled x1xb = DTM enabled xx1b = OTM enabled

60.6.1.4 ENGINE_x Development Status Register (NDEDI_ENGINE_x_DS)

The NDEDI_ENGINE_x_DS register shows the status of various conditions that impact development support. All status bits are dynamic and do not require clearing. If the Class 3 is enabled, any time the value of any bit in this register changes, a Debug Status Event is queued, and then will generate a Debug Status Message. If the Event Queue is not enabled to store snapshots while a DS event should be queued, then an error event is queued as soon as the Event Queue is enabled for storing snapshots.

DS register dynamically tracks the debug condition status even when several of them are simultaneously asserted. For instance, if EVTI and external debug are asserted at the same time, ETPU will enter in debug mode and NDEDI will assert the bits EBP(EVTI) and CBP. Then, if CBR is asserted the EBP bit will be cleared but the CBP bit will continue to be active since the external debug request is still on.

In some extreme cases NDEDI may not signalize more than one breakpoints condition at the same address. Please refer to [Multiple breakpoint conditions at the same address](#) for more details about this limitation.

Client Select: ENGINE_x_SRC

Register index: 4

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	HAT	IIBP	CBP	TBP	EBP	0	0	0	0	0	0	0	0
W																
RESET:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	BP							0	0	DBS	STP	HWE	HWB	SWB	SSS
W																
RESET:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

 = Reserved

Figure 60-4. ENGINE_x Development Status Register (NDEDI_ENGINE_x_DS)

Table 60-7. NDEDI_ENGINE_x_DS field descriptions

Field	Description
HAT	Halted with Active Thread The HAT bit indicates if the engine was halted during the execution of a thread or while it was IDLE. 0 = The ETPU engine is halted with no thread currently active 1 = The ETPU engine is halted in the middle of a thread execution

Table continues on the next page...

Table 60-7. NDEDI_ENGINEx_DS field descriptions (continued)

Field	Description
IIBP	Illegal Instruction Breakpoint Status The IIBP bit indicates if there is a breakpoint due to an illegal instruction execution causing the halt condition 0 = Breakpoint source is not an illegal instruction execution 1 = Breakpoint source is an illegal instruction execution
CBP	Client Breakpoint Status The CBP bit indicates if there is a breakpoint from an external client causing the halt condition 0 = Breakpoint source is not an external client 1 = Breakpoint source is an external client
TBP	Twin Breakpoint Status The TBP bit indicates if there is a breakpoint from the twin ETPU engine causing the halt condition 0 = Breakpoint source is not the twin ETPU engine 1 = Breakpoint source is the twin ETPU engine
EBP	EVTI Breakpoint Status The EBP bit indicates if there is a breakpoint caused by EVTI assertion 0 = Breakpoint source is not EVTI assertion 1 = Breakpoint source is EVTI assertion
BP	Internal Breakpoint Status The BP field shows which hardware breakpoints¹ have occurred causing the ENGINEx to enter debug mode . BP values: 0b0000000 = No Breakpoint Condition Occurred 0b1xxxxxx = ETPU Breakpoint 1 (based on BWC1 register) 0bx1xxxxx = ETPU Breakpoint 2 (based on BWC2 register) 0bxx1xxxx = Channel Register Write Breakpoint 0bxxx1xxx = Host Service Request Breakpoint 0bxxxx1xx = Link Register Breakpoint 0bxxxxx1x = MRL Breakpoint 0bxxxxxx1 = TDL Breakpoint
DBS	Debug Status The DBS bit indicates if ENGINEx is in debug mode. The fields HAT, IIBP, CBP, TBP, EBP, BP, HWB, SWB and SSS are only meaningful if the processor is indicated as halted. 0 = Processor not halted. 1 = Processor halted in debug mode

Table continues on the next page...

Table 60-7. NDEDI_ENGINEx_DS field descriptions (continued)

Field	Description
STP	Stop Status Indicates if the ENGINEx is in stop mode . Besides the IPI Green line stop mode mechanism, the ETPU engines may independently be programmed for STOP mode by asserting the MDIS bit of the ECR register. Thus, the STP field indicates if the engine is in stop mode no matter if due to the IPI Green line stop mode assertion or due to the MDIS field. Please refer to the ETPU BG for further information about the stop mode mechanisms. 0 = Processor not stopped 1 = Processor stopped in low power mode
HWE	Hardware Error Status Indicates if the ENGINEx has encountered a HW error . The only hardware error defined for the ENGINE is a MISC miscompare.Please refer to the ETPU BG for further information about MISC. 0 = No HW error 1 = Non-recoverable HW error occurred
HWB	Hardware Breakpoint Status Indicates if one of the hardware breakpoints described at the BP field has occurred . 0 = No hardware breakpoint occurred 1 = Hardware breakpoint occurred
SWB	Software Breakpoint Status Indicates if a software breakpoint has occurred. A software breakpoint occurs with the execution of the HALT instruction. For further information, please refer to the ETPU BG. 0 = No software breakpoint occurred 1 = Software breakpoint occurred
SSS	Single Step Status Indicates if the processor is halted for debug mode due to single step occurrence . 0 = Processor not halted due to single step 1 = Processor halted in debug mode due to single step

1. A hardware breakpoint is a breakpoint configured by BWC1, BWC2 or BWC3.

60.6.1.5 ENGINEx Watchpoint Trigger Register (NDEDI_ENGINEx_WT)

The NDEDI_ENGINEx_WT register allows traces to be enabled and/or disabled on the occurrence of a watchpoint.If the same watchpoint is programed to enable and disable a trace, the occurrence of the watchpoint toggles the current value of the trace flags If one watchpoint occurs to enable trace and a different watchpoint occurs to disable trace at the same time the current value of the trace flags will also be toggled.Trace enable flags from triggers (PTEF and DTEF) are ORed with the appropriate bit in the TM field of the DC register and sent to the trace blocks.

Client Select: ENGINEx_SRC

Register definition

Register index: 11

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	PTS			PTE			DTS			DTE			0	0	0	0
W																
RESET:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	PTEF	DTEF
W															PTEC	DTEC
RESET:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

= Reserved

Figure 60-5. ENGINEx Watchpoint Trigger Register (NDEDI_ENGINEx_WT)

Table 60-8. NDEDI_ENGINEx_WT field descriptions

Field	Description
PTS	Program Trace Start This bit allows program trace to be enabled at a watchpoint occurrence. PTS values: 0b000 = Trigger Disabled 0b001 = Program Trace enabled on ETPU Watchpoint 1 occurrence 0b010 = Program Trace enabled on ETPU Watchpoint 2 occurrence 0b011 = Program Trace enabled on Channel Register Write Watchpoint occurrence 0b100 = Program Trace enabled on Host Service Request Watchpoint occurrence 0b101 = Program Trace enabled on Link Register Watchpoint occurrence 0b110 = Program Trace enabled on MRL Watchpoint occurrence 0b111 = Program Trace enabled on TDL Watchpoint occurrence
PTE	Program Trace End This bit allows program trace to be disabled at a watchpoint occurrence. PTE values: 0b000 = Trigger Disabled 0b001 = Program Trace disabled on ETPU Watchpoint 1 occurrence 0b010 = Program Trace disabled on ETPU Watchpoint 2 occurrence 0b011 = Program Trace disabled on Channel Register Write Watchpoint occurrence 0b100 = Program Trace disabled on Host Service Request Watchpoint occurrence 0b101 = Program Trace disabled on Link Register Watchpoint occurrence 0b110 = Program Trace disabled on MRL Watchpoint occurrence 0b111 = Program Trace disabled on TDL Watchpoint occurrence

Table continues on the next page...

Table 60-8. NDEDI_ENGINEx_WT field descriptions (continued)

Field	Description
DTS	Data Trace Start This bit allows data trace to be enabled at a watchpoint occurrence. DTS values: 0b000 = Trigger Disabled 0b001 = Data Trace enabled on ETPU Watchpoint 1 occurrence 0b010 = Data Trace enabled on ETPU Watchpoint 2 occurrence 0b011 = Data Trace enabled on Channel Register Write Watchpoint occurrence 0b100 = Data Trace enabled on Host Service Request Watchpoint occurrence 0b101 = Data Trace enabled on Link Register Watchpoint occurrence 0b110 = Data Trace enabled on MRL Watchpoint occurrence 0b111 = Data Trace enabled on TDL Watchpoint occurrence
DTE	Data Trace End This bit allows data trace to be disabled at a watchpoint occurrence. DTE values: 0b000 = Trigger Disabled 0b001 = Data Trace disabled on ETPU Watchpoint 1 occurrence 0b010 = Data Trace disabled on ETPU Watchpoint 2 occurrence 0b011 = Data Trace disabled on Channel Register Write Watchpoint occurrence 0b100 = Data Trace disabled on Host Service Request Watchpoint occurrence 0b101 = Data Trace disabled on Link Register Watchpoint occurrence 0b110 = Data Trace disabled on MRL Watchpoint occurrence 0b111 = Data Trace disabled on TDL Watchpoint occurrence
PTEF	Program Trace Enable Flag This bit shows if program trace is currently enabled due to a watchpoint trigger. 0 = Program Trace not enabled due to watchpoint trigger 1 = Program Trace enabled due to watchpoint trigger
PTEC	Program Trace Enable Clear Writing a one to the PTEC bit clears the PTEF flag. This allows the development tool to turn off program trace started via a trigger without using a second trigger or module reset. 0 = Keep PTEF flag unaltered 1 = Clear the PTEF flag
DTEF	Data Trace Enable Flag This bit shows if data trace is currently enabled due to a watchpoint trigger. 0 = Data Trace not enabled due to watchpoint trigger 1 = Data Trace enabled due to watchpoint trigger

Table continues on the next page...

Table 60-8. NDEDI_ENGINEx_WT field descriptions (continued)

Field	Description
DTEC	Data Trace Enable Clear Writing a one to the DTEC bit clears the DTEF flag. This allows the development tool to turn off data trace started via a trigger without using a second trigger or module reset. 0 = Keep DTEF flag unaltered 1 = Clear the DTEF flag

60.6.1.6 ENGINEx Data Trace Control Register (NDEDI_ENGINEx_DTC)

The NDEDI_ENGINEx_DTC register controls which address ranges are enabled for data trace, and if reads and/or writes are traced in that range.

Client Select: ENGINEx_SRC

Register index: 13

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	RWT0		RWT1		RWT2		RWT3		0	0	0	0	0	0	0	0
W																
RESET:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	0	0	0	0	0	RC0	RC1	RC2	RC3	0	0	0	0
W																
RESET:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

= Reserved

Figure 60-6. ENGINEx Data Trace Control Register (NDEDI_ENGINEx_DTC)**Table 60-9. NDEDI_ENGINEx_DTC field descriptions**

Field	Description
RWT0	ENGINEx Read/Write Trace 0 Control RWT0 controls whether data trace messages are generated for ENGINEx accesses inside ETPU data trace window 0 (see Data Trace Address Range 0 Register (NDEDI_DTAR0)), and if so, whether reads, writes, or both generate the data trace messages. RWT0 values: 0b00 = No ENGINEx data trace messages generated for ETPU window 0 0b01 = Enable ENGINEx data read trace for ETPU window 0 0b10 = Enable ENGINEx data write trace for ETPU window 0 00b11 = Enable ENGINEx data read and write trace for ETPU window 0

Table continues on the next page...

Table 60-9. NDEDI_ENGINEx_DTC field descriptions (continued)

Field	Description
RWT1	ENGINEx Read/Write Trace 1 Control RWT1 controls whether data trace messages are generated for ENGINEx accesses inside ETPU data trace window 1 (see Data Trace Address Range 1 Register (NDEDI_DTAR1)), and if so, whether reads, writes, or both generate the data trace messages. RWT1 values: 0b00 = No ENGINEx data trace messages generated for ETPU window 1 0b01 = Enable ENGINEx data read trace for ETPU window 1 0b10 = Enable ENGINEx data write trace for ETPU window 1 0b11 = Enable ENGINEx data read and write trace for ETPU window 1
RWT2	ENGINEx Read/Write Trace 2 Control RWT2 controls whether data trace messages are generated for ENGINEx accesses inside ETPU data trace window 2 (see Data Trace Address Range 2 Register (NDEDI_DTAR2)), and if so, whether reads, writes, or both generate the data trace messages. RWT2 values: 0b00 = No ENGINEx data trace messages generated for ETPU window 2 0b01 = Enable ENGINEx data read trace for ETPU window 2 0b10 = Enable ENGINEx data write trace for ETPU window 2 0b11 = Enable ENGINEx data read and write trace for ETPU window 2
RWT3	ENGINEx Read/Write Trace 3 Control RWT3 controls whether data trace messages are generated for ENGINEx accesses inside ETPU data trace window 3 (see Data Trace Address Range 3 Register (NDEDI_DTAR3)), and if so, whether reads, writes, or both generate the data trace messages. RWT3 values: 0b00 = No ENGINEx data trace messages generated for ETPU window 3 0b01 = Enable ENGINEx data read trace for ETPU window 3 0b10 = Enable ENGINEx data write trace for ETPU window 3 0b11 = Enable ENGINEx data read and write trace for ETPU window 3
RC0	Range Control 0 Controls which addresses match data trace window 0 . 0 = Trace address inside (inclusive) of data trace window 0 1 = Trace addresses outside (exclusive) of data trace window 0
RC1	Range Control 1 Controls which addresses match data trace window 1 . 0 = Trace address inside (inclusive) of data trace window 1 1 = Trace addresses outside (exclusive) of data trace window 1
RC2	Range Control 2 Controls which addresses match data trace window 2 . 0 = Trace address inside (inclusive) of data trace window 2 1 = Trace addresses outside (exclusive) of data trace window 2

Table continues on the next page...

Table 60-9. NDEDI_ENGINEx_DTC field descriptions (continued)

Field	Description
RC3	Range Control 3 Controls which addresses match data trace window 3 . 0 = Trace address inside (inclusive) of data trace window 3 1 = Trace addresses outside (exclusive) of data trace window 3

60.6.1.7 ENGINEx Breakpoint/Watchpoint Control 1/2 Register (NDEDI_ENGINEx_BWC1/2)

The ENGINEx Breakpoint/Watchpoint Control 1 Register is used to configure ENGINEx breakpoint/watchpoint 1 along with the NDEDI_ENGINEx_BWA1 and NDEDI_ENGINEx_BWD1 registers.

The ENGINEx Breakpoint/Watchpoint Control 2 Register is used to configure ENGINEx breakpoint/watchpoint 2 along with the NDEDI_ENGINEx_BWA2 and NDEDI_ENGINEx_BWD2 registers.

These registers are responsible for configuring the hardware breakpoints/watchpoints 1 and 2. And every field of this register is related to these breakpoints only. When a field is set to a reserved value the behavior defaults to the reset values behavior.

The register can be configured for breakpoint/watchpoint based on data write values, instruction address and data read or write addresses.

When a hardware breakpoint occurs the program execution is halted before the instruction is executed. This means that data access breakpoints occur before the SPRAM access actually takes place.

The data accesses that occur within the Time Slot Transition (TST) do not generate a hardware breakpoint. Thus, it is not expected that the ETPU system signalize these transactions to the NDEDI interface.

Client Select: ENGINEx_SRC

Register index: 22

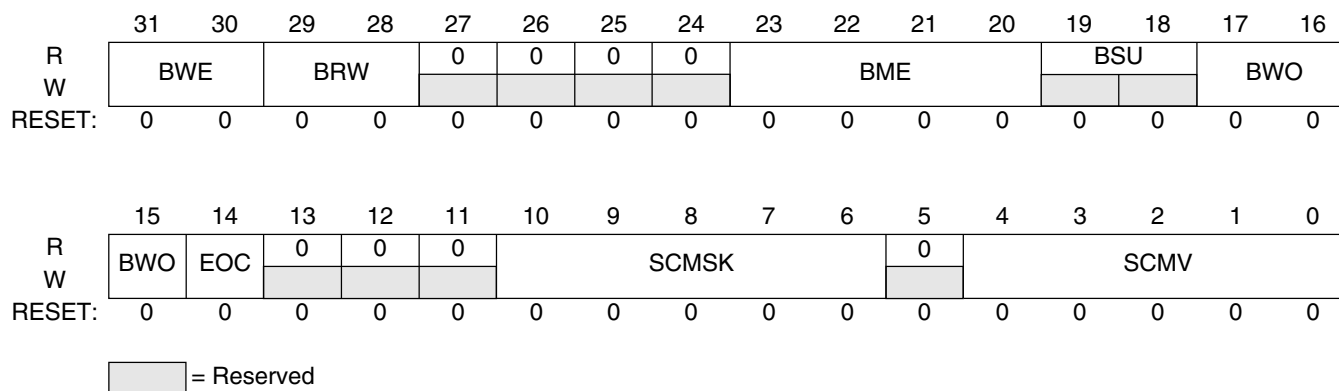


Figure 60-7. ENGINE_x Breakpoint/Watchpoint Control 1 Register (NDEDI_ENGINE_x_BWC1)

Client Select: ENGINE_x_SRC

Register index: 23

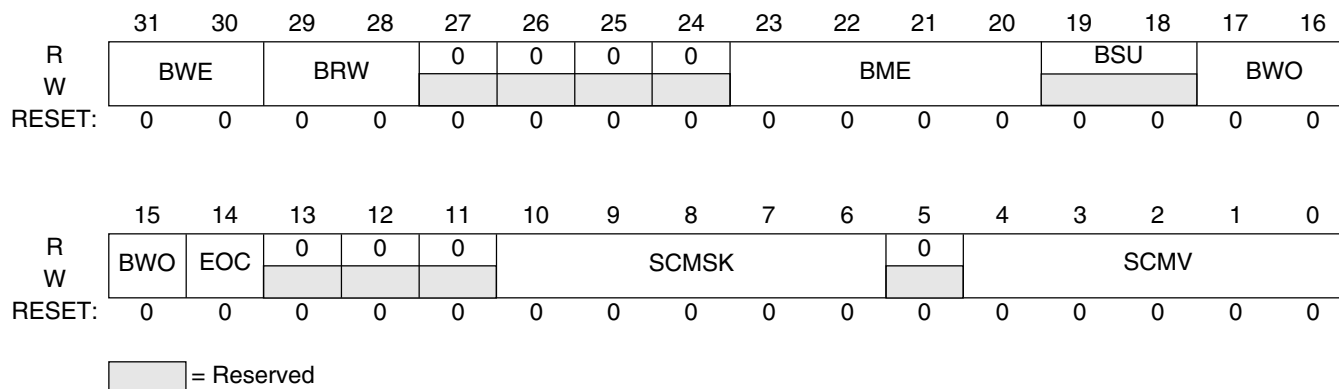


Figure 60-8. ENGINE_x Breakpoint/Watchpoint Control 2 Register (NDEDI_ENGINE_x_BWC2)

Table 60-10. NDEDI_ENGINEx_BWC1/2 field descriptions

Field	Description
BWE	Breakpoint/Watchpoint Enable The BWE field is used to enable the ENGINEx hardware breakpoint or watchpoint. BWE values: 0b00 = Disabled 0b01 = Breakpoint Enabled 0b10 = Reserved 0b11 = Watchpoint Enabled Changing BWE from 'Breakpoint Enabled'(BWE=0b01) to 'Watchpoint Enabled'(BWE=0b11) and vice-versa must only be done while ETPU is either halted or stopped. To set BWE as either 'Breakpoint Enabled'(BWE=0b01) or 'Watchpoint Enabled'(BWE=0b11) while ETPU is neither halted nor stopped the user must previously set BWE as 'Disabled' (BWE=0b00). All other fields of this register must only be changed when BWE=0b00 (disabled) when the engine is not stopped or halted, in 3 steps: (a) write BWE=0b00; (b) change other fields; (c) write BWE with a valid non-zero value, keeping the other fields unchanged.
BRW	Breakpoint/Watchpoint Read/Write Select The BRW field selects if read and/or write accesses cause a hardware breakpoint/watchpoint. BRW values: 0b00 = Match on read SPRAM access 0b01 = Match on write SPRAM access 0b10 = Match on any SPRAM accesses 0b11 = Reserved
BME	Breakpoint/Watchpoint Data Mask Enable The BME field selects which data bytes are used for hardware breakpoint/watchpoint generation. BME values: 0b0000 = All bytes of data compared 0b1xxx = Most significant byte of data masked out ... 0bxxx1 = Least significant byte of data masked out
BSU	Breakpoint/Watchpoint Data Size Unit The BSU is a read-only field to indicate the data size unit used by the NDEDI block. This field always reads as 0b00 to indicate the data size unit is 1 byte.

Table continues on the next page...

Table 60-10. NDEDI_ENGINEx_BWC1/2 field descriptions (continued)

Field	Description
BWO	Breakpoint/Watchpoint Operand The BWO selects if address and/or data matching is done and if matching is done on data fetches or instruction fetches. BWO values: 0b00x = Breakpoint/Watchpoint Disabled 0bx10 = Reserved 0b011 = Compare data of a SPRAM write access with BWD 0b100 = Compare instruction Address with BWA 0b101 = Compare Address of a SPRAM write/read access with BWA 0b111 = Compare Data and Address of a SPRAM write access with BWD and BWA respectively
EOC	$\overline{\text{EVTO}}$ Control The EOC bit controls the action of the $\overline{\text{EVTO}}$ output at the occurrence of a hardware breakpoint or watchpoint. 0 = Breakpoint/Watchpoint status indication is not output on $\overline{\text{EVTO}}$ 1 = Breakpoint/Watchpoint status indication is output on $\overline{\text{EVTO}}$
SCMSK	Serviced Channel Mask Value The SCMSK field is used to mask what bits of the SCMV field are compared to the serviced channel. When a bit in the SCMSK field is 0, that bit is not compared for masking purposes. serviced channel match = (SCMV&SCMSK) == (Serviced Channel & SCMSK);
SCMV	Serviced Channel Match Value The SCMV field value is compared against the ENGINEx serviced channel when generating the breakpoints and watchpoints.

60.6.1.8 ENGINEx Breakpoint/Watchpoint Address 1/2 Register (NDEDI_ENGINEx_BWA1/2)

The ENGINEx Breakpoint/Watchpoint Address 1 Register is used to configure ENGINEx hardware breakpoint/watchpoint 1 along with the NDEDI_ENGINEx_BWC1 and NDEDI_ENGINEx_BWD1 registers. It is used for data access and instruction fetches breakpoints.

The ENGINEx Breakpoint/Watchpoint Address 2 Register is used to configure ENGINEx hardware breakpoint/watchpoint 2 along with the NDEDI_ENGINEx_BWC2 and NDEDI_ENGINEx_BWD2 registers. It is used for data access and instruction fetches breakpoints.

Client Select: ENGINEx_SRC

Register index: 30

Register definition

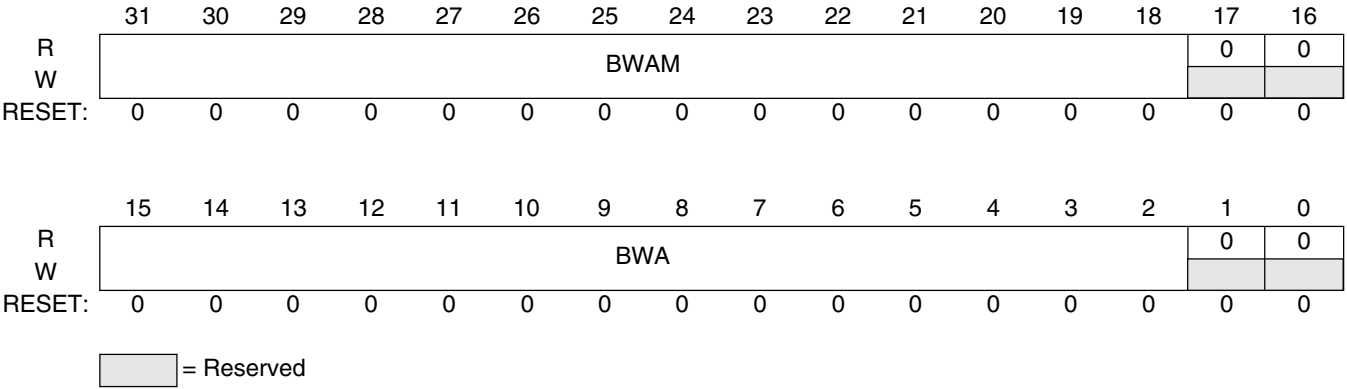


Figure 60-9. ENGINE_x Breakpoint/Watchpoint Address 1 Register (NDEDI_ENGINE_x_BWA1)

Client Select: ENGINE_x_SRC

Register index: 31

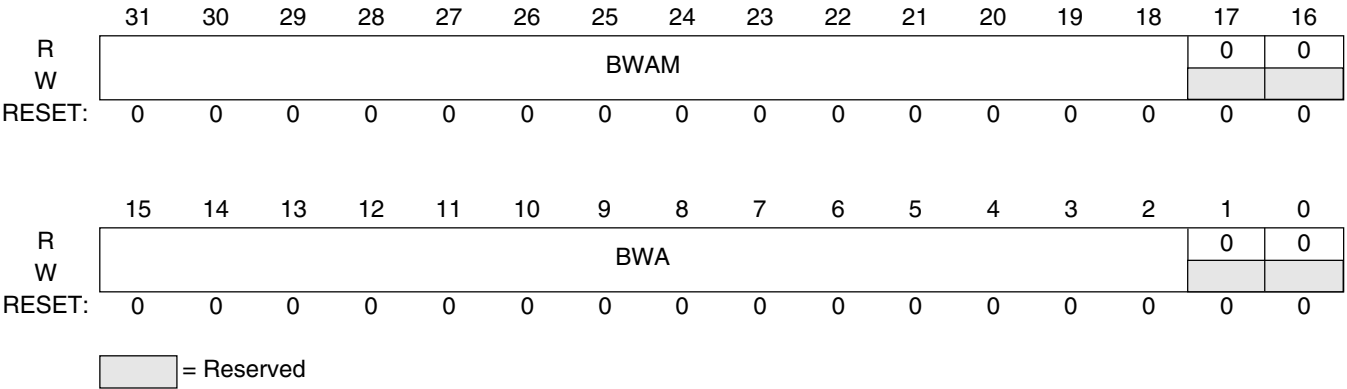


Figure 60-10. ENGINE_x Breakpoint/Watchpoint Address 2 Register (NDEDI_ENGINE_x_BWA2)

Table 60-11. NDEDI_ENGINE_x_BWA1/2 field descriptions

Field	Description
BWAM	Breakpoint/Watchpoint Address Mask The BWAM field is used to select which address bits are compared for breakpoint and watchpoint generation. address match = (BWA&BWAM) == (Address & BWAM);
BWA	Breakpoint/Watchpoint Address The BWA field is used to compare address operands (address of instruction or data). The BWA field represent a word addressable address. The most significant two bits are ignored for address of data comparisons because the SPRAM address space is only 14 bits.

60.6.1.9 ENGINE_x Breakpoint/Watchpoint Data 1/2 Register (NDEDI_ENGINE_x_BWD1/2)

The ENGINE_x Breakpoint/Watchpoint Data 1 Register is used to configure the ENGINE_x hardware breakpoint/watchpoint 1 along with the NDEDI_ENGINE_x_BWC1 and NDEDI_ENGINE_x_BWA1 registers. It is used for data access breakpoints only.

The ENGINE_x Breakpoint/Watchpoint Data 2 Register is used to configure the ENGINE_x hardware breakpoint/watchpoint 2 along with the NDEDI_ENGINE_x_BWC2 and NDEDI_ENGINE_x_BWA2 registers. It is used for data access breakpoints only.

Client Select: ENGINE_x_SRC

Register index: 38

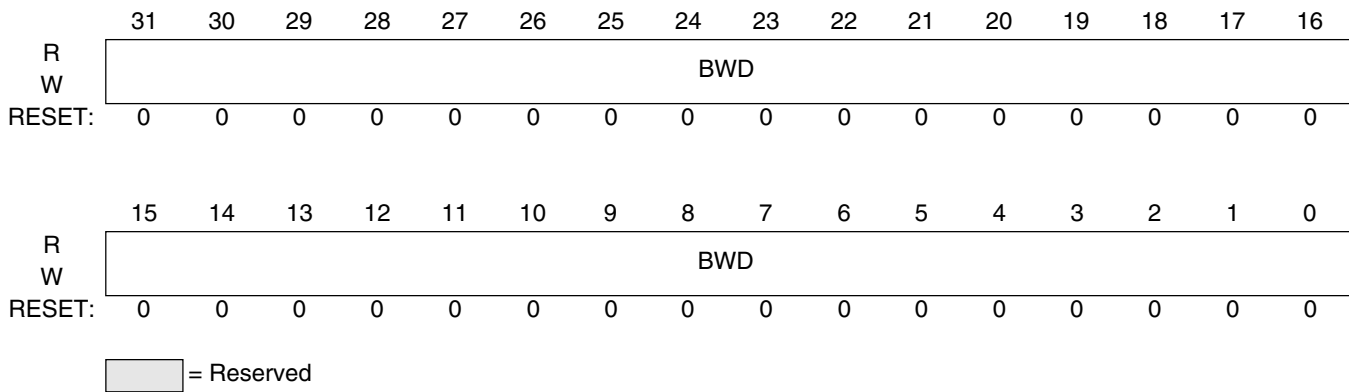


Figure 60-11. ENGINE_x Breakpoint/Watchpoint Data 1 Register (NDEDI_ENGINE_x_BWD1)

Client Select: ENGINE_x_SRC

Register index: 39

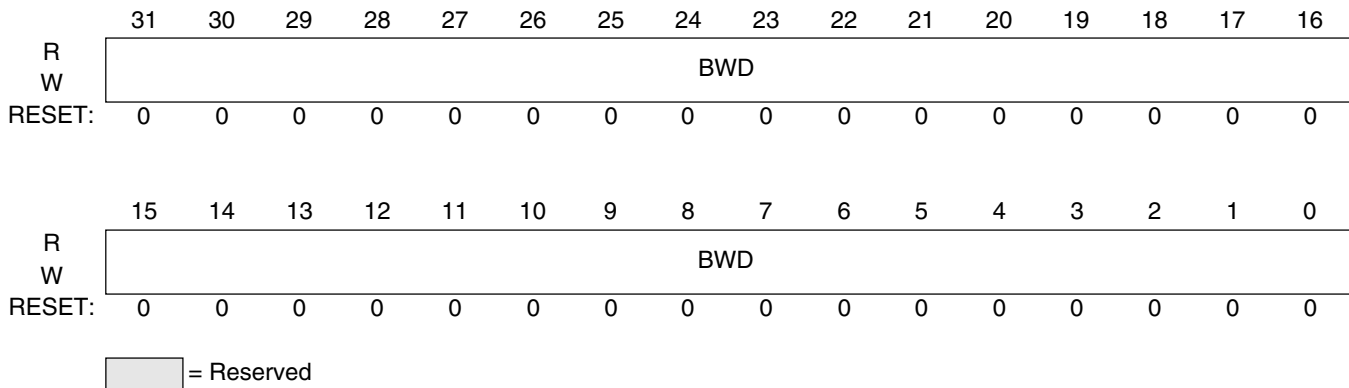


Figure 60-12. ENGINE_x Breakpoint/Watchpoint Data 2 Register (NDEDI_ENGINE_x_BWD2)

Table 60-12. NDEDI_ENGINEx_BWD1/2 field descriptions

Field	Description
BWD	Breakpoint/Watchpoint Data The BWD field is used to compare data operands whether in a SPRAM write access. Depending on the size of the access, only certain bits of BWD are compared against the data written to the SPRAM. Table 60-13 shows which bits of BWD are compared against the data written to the SPRAM.

Table 60-13. Bits compared against BWD

Operation size	Bits compared
8-bits	8 most significant bits of BWD
24-bits	24 least significant bits of BWD
32-bits	All bits from BWD

60.6.1.10 ENGINEx Program Trace Channel Enable Register (NDEDI_ENGINEx_PTCE)

The NDEDI_ENGINEx_PTCE register enables program tracing for each of the 32 ENGINEx channels. There is one enable bit per channel. And the bit corresponding to the serviced channel is the only bit tested for enabling program trace. For these bits to have any effect, program trace must also be enabled in the TM field of the DC register or via a watchpoint trigger.

Client Select: ENGINEx_SRC

Register index: 64

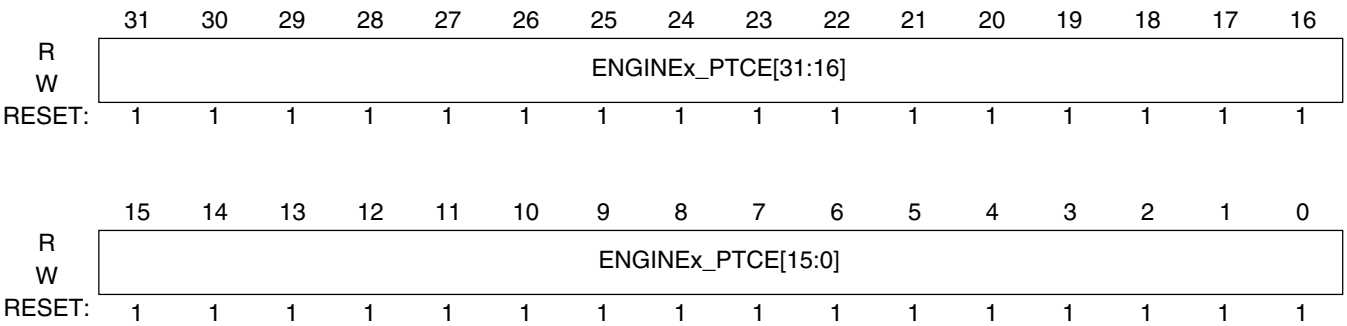


Figure 60-13. ENGINEx Program Trace Channel Enable Register (NDEDI_ENGINEx_PTCE)

Table 60-14. NDEDI_ENGINEx_PTCE field descriptions

Field	Description
ENGINEx_PTCEy	ENGINEx Timer Channel y Program Trace Enable 0 = Program tracing disabled for ENGINEx timer channel y 1 = Program tracing enabled for ENGINEx timer channel y

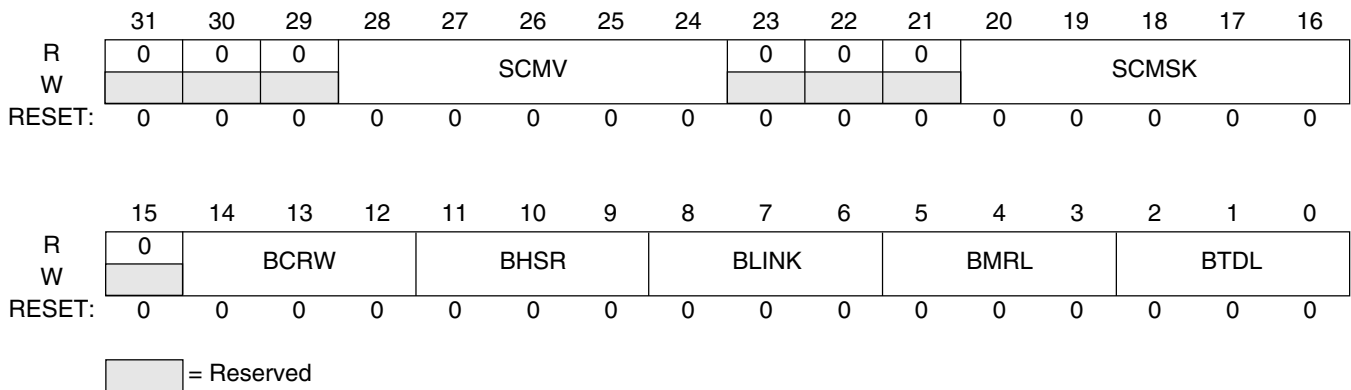
60.6.1.11 ENGINEx Breakpoint/Watchpoint Control 3 Register (NDEDI_ENGINEx_BWC3)

The ENGINEx Breakpoint/Watchpoint Control 3 Register is used to configure miscellaneous ENGINEx hardware breakpoint/watchpoint sources. This is a vendor-defined register with different fields than BWC1 and BWC2.

When a field is set to a reserved value the behavior defaults to the reset values behavior.

Client Select: ENGINEx_SRC

Register index: 24

**Figure 60-14. ENGINEx Breakpoint/Watchpoint Control 3 Register (NDEDI_ENGINEx_BWC3)****Table 60-15. NDEDI_ENGINEx_BWC3 field descriptions**

Field	Description
SCMV	Serviced Channel Match Value The SCMV field value is compared against the serviced channel register when generating a breakpoint due to a write in the CHAN register or due to the start of a service.
SCMSK	Serviced Channel Mask Value The CMSK field is used to mask what bits of the SCMV field are compared to the serviced channel register when generating a breakpoint due to a write in the CHAN register or due to the start of a service. When a bit in the SCMSK field is 0, that bit is not compared for masking purposes.

Table continues on the next page...

Table 60-15. NDEDI_ENGINEx_BWC3 field descriptions (continued)

Field	Description
BCRW	Break on Channel Register Write The BCRW field configures the action when there is a write to the CHAN register and there is a serviced channel match (as described bellow). The breakpoint causes execution to halt before the CHAN register is written. serviced channel match = (SCMV&SCMSK) == (Serviced Channel & SCMSK); BCRW values: 0b00x = Disabled 0b01x = Breakpoint Enabled 0b10x = Reserved 0b11x = Watchpoint Enabled 0bxx0 = Breakpoint/Watchpoint status indication not output on $\overline{\text{EVTO}}$ 0bxx1 = Breakpoint/Watchpoint status indication output on $\overline{\text{EVTO}}$
BHSR	Break on Host Service Request The BHSR field configures the action when there is a serviced channel match (as described bellow) and the host service request is asserted at the beginning of the channel service. The breakpoint causes execution to halt after the time slot transition (TST) completes but before the first instruction of the thread is executed. serviced channel match = (SCMV&SCMSK) == (Serviced Channel & SCMSK); BHSR values: 0b00x = Disabled 0b01x = Breakpoint Enabled 0b10x = Reserved 0b11x = Watchpoint Enabled 0bxx0 = Breakpoint/Watchpoint status indication not output on $\overline{\text{EVTO}}$ 0bxx1 = Breakpoint/Watchpoint status indication output on $\overline{\text{EVTO}}$
BLINK	Break on Link Service Request The BLINK field configures the action when there is a serviced channel match (as described bellow) and the link service register is asserted at the beginning of the channel service. The breakpoint causes execution to halt after the time slot transition (TST) completes but before the first instruction of the thread is executed. serviced channel match = (SCMV&SCMSK) == (Serviced Channel & SCMSK); BLINK values: 0b00x = Disabled 0b01x = Breakpoint Enabled 0b10x = Reserved 0b11x = Watchpoint Enabled 0bxx0 = Breakpoint/Watchpoint status indication not output on $\overline{\text{EVTO}}$ 0bxx1 = Breakpoint/Watchpoint status indication output on $\overline{\text{EVTO}}$

Table continues on the next page...

Table 60-15. NDEDI_ENGINEx_BWC3 field descriptions (continued)

Field	Description
BMRL	Break on Match Recognition Request The BMRL field configures the action when there is a serviced channel match (as described bellow) and either MRL1 or MRL2 is asserted at the beginning of the channel service. The breakpoint causes execution to halt after the time slot transition (TST) completes but before the first instruction of the thread is executed. serviced channel match = (SCMV&SCMSK) == (Serviced Channel & SCMSK); BMRL values: 0b00x = Disabled 0b01x = Breakpoint Enabled 0b10x = Reserved 1 0b11x = Watchpoint Enabled 0bxx0 = Breakpoint/Watchpoint status indication not output on $\overline{\text{EVTO}}$ 0bxx1 = Breakpoint/Watchpoint status indication output on $\overline{\text{EVTO}}$
BTDL	Break on Transition Detect Request The BTDL field configures the action when there is a serviced channel match (as described bellow) and the TDL1 or TDL2 is asserted at the beginning of the channel service. The breakpoint causes execution to halt after the time slot transition (TST) completes but before the first instruction of the thread is executed. serviced channel match = (SCMV&SCMSK) == (Serviced Channel & SCMSK); BTDL values: 0b00x = Disabled 0b01x = Breakpoint Enabled 0b10x = Reserved 0b11x = Watchpoint Enabled 0bxx0 = Breakpoint/Watchpoint status indication not output on $\overline{\text{EVTO}}$ 0bxx1 = Breakpoint/Watchpoint status indication output on $\overline{\text{EVTO}}$

60.6.1.12 ENGINEx Microinstruction Debug Register (NDEDI_ENGINEx_INST)

The NDEDI_ENGINEx_INST register is used for forcing a microinstruction on the microinstruction register. This register must only be accessed while the ENGINEx is halted. Writes to this register are ignored if accessed while the ENGINEx is not halted.

Client Select: ENGINEx_SRC

Register index: 69

Register definition

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W	ENGINE _x _INST															
RESET:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W	ENGINE _x _INST															
RESET:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 60-15. ENGINE_x Microinstruction Debug Register (NDEDI_ENGINE_x_INST)

Writes to the register in debug mode force the execution of an arbitrary instruction. This is used to retrieve information about internal status, set internal flag values among other things during debug mode. Some fields of this microinstruction may be ignored, for instance an END instruction will be ignored. For further information about the arbitrary instruction refer to the ETPU BG Section 4.10.2.7.

Reads from the register returns 0.

Table 60-16. NDEDI_ENGINE_x_INST field descriptions

Field	Description
ENGINE _x _INST	the microinstruction forced into the microinstruction register

60.6.1.13 ENGINE_x Microprogram Counter Debug Register (NDEDI_ENGINE_x_MPC)

The NDEDI_ENGINE_x_MPC register is used for reading the microprogram counter value. This register is only meaningful while the ENGINE_x is halted. If accessed while the ENGINE_x is not halted, reads return zero. Writes to this register are always ignored.

Client Select: ENGINE_x_SRC

Register index: 70

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
RESET:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R																
W																
RESET:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 60-16. ENGINE_x Microprogram Counter Debug Register (NDEDI_ENGINE_x_MPC)

Table 60-17. NDEDI_ENGINEx_MPC field descriptions

Field	Description
NDEDI_ENGINEx_MPC	Microprogram counter value

60.6.1.14 ENGINEx Channel Flag Status Register (NDEDI_ENGINEx_CFSR)

The NDEDI_ENGINEx_CFSR register provides read-only access to the flags of the channel being serviced by ENGINEx. Reads to this register when ENGINEx is out of halt mode return zero., Transition Detection and Match Recognition flags are available in two versions: One reflecting the Branch Condition Selection (BCC), TDL1/2 and MRL1/2, and an ‘internal’ version, ITDL1/2 and IMR1/2, reflecting the flag value within the channel logic.

Client Select: ENGINEx_SRC

Register index: 71

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	V	N	C	Z	MV	MN	MC	MZ	TDL1	TDL2	MRL1	MRL2	LSR	SML-CK	FM	
W																
RESET:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	SRI	HSR			FLAG 0	FLAG 1	MRLE 1	MRLE 2	ITDL1	ITDL2	IMRL 1	IMRL 2	PSS	PSTI	PSTO	OBE
W																
RESET:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 60-17. ENGINEx Channel Flag Status Register (NDEDI_ENGINEx_CSFR)**Table 60-18. NDEDI_ENGINEx_CFSR field descriptions**

Field	Description
V	EAU Overflow Flag Overflow status flag from an EAU operation. 0 = EAU overflow flag is cleared 1 = EAU overflow flag is set
N	EAU Negative Flag Negative status flag from an EAU operation. 0 = EAU negative flag is cleared 1 = EAU negative flag is set

Table continues on the next page...

**Table 60-18. NDEDI_ENGINEx_CFSR field descriptions
(continued)**

Field	Description
C	EAU Carry Flag Carry status flag from an EAU operation. 0 = EAU carry flag is cleared 1 = EAU carry flag is set
Z	EAU Carry Flag Zero status flag from an EAU operation. 0 = EAU zero flag is cleared 1 = EAU zero flag is set
MV	MAC/Divide Unit Overflow Flag Overflow status flag from an MAC/Divide Unit operation. 0 = MAC/Divide Unit overflow flag is cleared 1 = MAC/Divide Unit overflow flag is set
MN	MAC/Divide Unit Negative Flag Negative status flag from an MAC/Divide Unit operation. 0 = MAC/Divide Unit negative flag is cleared 1 = MAC/Divide Unit negative flag is set
MC	MAC/Divide Unit Carry Flag Carry status flag from an MAC/Divide Unit operation. 0 = MAC/Divide Unit carry flag is cleared 1 = MAC/Divide Unit carry flag is set
MZ	MAC/Divide Unit Carry Flag Zero status flag from an MAC/Divide Unit operation. 0 = MAC/Divide Unit zero flag is cleared 1 = MAC/Divide Unit zero flag is set
TDL1	Channel Transition Detection Latch 1 Channel Transition Detection status flag 1 as available in the BCC. 0 = Channel Transition flag 1 in BCC is cleared 1 = Channel Transition flag 1 in BCC is set
TDL2	Channel Transition Detection Latch 2 Channel Transition Detection status flag 2 as available in the BCC. 0 = Channel Transition flag 2 in BCC is cleared 1 = Channel Transition flag 2 in BCC is set
MRL1	Channel Match Recognition Latch 1 Channel Match Recognition status flag 1 as available in the BCC. 0 = Match Recognition flag 1 in BCC is cleared 1 = Match Recognition flag 1 in BCC is set

Table continues on the next page...

**Table 60-18. NDEDI_ENGINEx_CFSR field descriptions
(continued)**

Field	Description
MRL2	Channel Match Recognition Latch 2 Channel Match Recognition status flag 2 as available in the BCC. 0 = Match Recognition flag 2 in BCC is cleared 1 = Match Recognition flag 2 in BCC is set
LSR	Channel Link Service Register Channel Link Service Register flag. 0 = Link even to currently serviced channel did not occur 1 = Link event to currently serviced channel occurred
SMLCK	Channel Semaphore Flag Channel Semaphore flag. 0 = Semaphore flag is cleared 1 = Semaphore flag is set
FM	Channel Function Mode Reflects the state of the FM field of the Status Control Register of the current channel.
SRI	Service Request Inhibit Flag that blocks channel service requests due to the assertion of MRL1/2 and/or TDL1/2. 0 = Service requests not inhibited 1 = Service requests inhibited
HSR	Host Service Request Indicates the Entry Point HSR of the thread currently executing, if any. This was the value of HSR of the serviced channel when the entry point was chosen. It is not necessarily the value of the Host Service Request Register of the current channel.
FLAG0	Channel State Resolution Flag 0 One of two flags per channel that can be set or cleared by microcode to further resolve the entry point for a channel service. 0 = FLAG0 was cleared by microcode 1 = FLAG0 was set by microcode
FLAG1	Channel State Resolution Flag 1 One of two flags per channel that can be set or cleared by microcode to further resolve the entry point for a channel service. 0 = FLAG1 was cleared by microcode 1 = FLAG1 was set by microcode
MRLE1	Channel Match Recognition Latch Enable 1 Channel Match Recognition Latch Enable 1. 0 = Match recognition disabled for event 1 1 = Match recognition enabled for event 1

Table continues on the next page...

**Table 60-18. NDEDI_ENGINEx_CFSR field descriptions
(continued)**

Field	Description
MRLE2	Channel Match Recognition Latch Enable 2 Channel Match Recognition Latch Enable 2. 0 = Match recognition disabled for event 2 1 = Match recognition enabled for event 2
ITDL1	Internal Channel Transition Detection Latch 1 Channel Transition Detection status flag 1 reflecting its status directly in the channel logic. 0 = Channel Transition flag 1 within the channel logic is cleared 1 = Channel Transition flag 1 within the channel logic is set
ITDL2	Internal Channel Transition Detection Latch 2 Channel Transition Detection status flag 2 reflecting its status directly in the channel logic. 0 = Channel Transition flag 2 within the channel logic is cleared 1 = Channel Transition flag 2 within the channel logic is set
IMRL1	Internal Channel Match Recognition Latch 1 Channel Match Recognition status flag 1 reflecting its status directly in the channel logic. 0 = Match Recognition flag 1 within the channel logic is cleared 1 = Match Recognition flag 1 within the channel logic is set
IMRL2	Internal Channel Match Recognition Latch 2 Channel Match Recognition status flag 2 reflecting its status directly in the channel logic. 0 = Match Recognition flag 2 within the channel logic is cleared 1 = Match Recognition flag 2 within the channel logic is set
PSS	Pin Sampled State Pin Sampled State reflects the status of the PSS flag for the current channel. 0 = PSS is cleared 1 = PSS is set
PSTI	Pin State Input Pin State Input reflects the status of the filtered input signal for the current channel. 0 = Filtered input signal is cleared 1 = Filtered input signal is set
PSTO	Pin State Output reflects the status of the value driven by the output control logic of the current channel. 0 = Output pin state register is cleared 1 = Output pin state register is set
OBE	Output Buffer Enable Output Buffer Enable reflects the status of the OBE control register for the current channel. 0 = Output Buffer Enable is cleared 1 = Output Buffer Enable is set

60.6.1.15 CDC Data Trace Control Register (NDEDI_CDC_DTC)

The NDEDI_CDC_DTC register controls which address ranges are enabled for CDC access data trace, and if reads and/or writes are traced in that range.

Client Select: CDC_SRC

Register index: 13

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	RWT0		RWT1		RWT2		RWT3		0	0	0	0	0	0	0	0
W																
RESET:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	0	0	0	0	0	RC0	RC1	RC2	RC3	0	0	0	0
W																
RESET:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

 = Reserved

Figure 60-18. CDC Data Trace Control Register (NDEDI_CDC_DTC)

Table 60-19. NDEDI_CDC_DTC field description

Field	Description
RWT0	CDC Read/Write Trace 0 Control RWT0 controls whether data trace messages are generated for CDC accesses inside ETPU data trace window 0 (see Data Trace Address Range 0 Register (NDEDI_DTAR0)), and if so, whether reads, writes, or both generate the data trace messages. RWT0 values: 0b00 = No CDC data trace messages generated for ETPU window 0 0b01 = Enable CDC data read trace for ETPU window 0 0b10 = Enable CDC data write trace for ETPU window 0 0b11 = Enable CDC data read and write trace for ETPU window 0
RWT1	CDC Read/Write Trace 1 Control RWT1 controls whether data trace messages are generated for CDC accesses inside ETPU data trace window 1 (see Data Trace Address Range 1 Register (NDEDI_DTAR1)), and if so, whether reads, writes, or both generate the data trace messages. RWT1 values: 0b00 = No CDC data trace messages generated for ETPU window 1 0b01 = Enable CDC data read trace for ETPU window 1 0b10 = Enable CDC data write trace for ETPU window 1 0b11 = Enable CDC data read and write trace for ETPU window 1

Table continues on the next page...

Table 60-19. NDEDI_CDC_DTC field description (continued)

Field	Description
RWT2	CDC Read/Write Trace 2 Control RWT2 controls whether data trace messages are generated for CDC accesses inside ETPU data trace window 2 (see Data Trace Address Range 2 Register (NDEDI_DTAR2)), and if so, whether reads, writes, or both generate the data trace messages. RWT2 values: 0b00 = No CDC data trace messages generated for ETPU window 2 0b01 = Enable CDC data read trace for ETPU window 2 0b10 = Enable CDC data write trace for ETPU window 2 0b11 = Enable CDC data read and write trace for ETPU window 2
RWT3	CDC Read/Write Trace 3 Control RWT3 controls whether data trace messages are generated for CDC accesses inside ETPU data trace window 3 (see Data Trace Address Range 3 Register (NDEDI_DTAR3)), and if so, whether reads, writes, or both generate the data trace messages. RWT3 value: 0b00 = No CDC data trace messages generated for ETPU window 3 0b01 = Enable CDC data read trace for ETPU window 3 0b10 = Enable CDC data write trace for ETPU window 3 0b11 = Enable CDC data read and write trace for ETPU window 3
RC0	Range Control 0 Controls which addresses match data trace window 0 . 0 = Trace address inside (inclusive) of data trace window 0 1 = Trace addresses outside (exclusive) of data trace window 0
RC1	Range Control 1 Controls which addresses match data trace window 1 . 0 = Trace address inside (inclusive) of data trace window 1 1 = Trace addresses outside (exclusive) of data trace window 1
RC2	Range Control 2 Controls which addresses match data trace window 2 . 0 = Trace address inside (inclusive) of data trace window 2 1 = Trace addresses outside (exclusive) of data trace window 2
RC3	Range Control 3 Controls which addresses match data trace window 3 . 0 = Trace address inside (inclusive) of data trace window 3 1 = Trace addresses outside (exclusive) of data trace window 3

60.6.1.16 Data Trace Address Range 0 Register (NDEDI_DTAR0)

The NDEDI_DTAR0 register defines a data trace address range. The address range is shared by the two ENGINES and the CDC. All three of these clients can independently enable the range for reads and/or writes. If the start address value is greater than the end address value, all accesses are considered to have the address value outside to the data trace window 0.²

Notice that the combination of the start and end addresses for data trace into one register does not follow the standard's recommended method of having separate registers for start and end addresses.

Client Select: ENGINE1_SRC or ENGINE2_SRC or CDC_SRC

Register index: 65

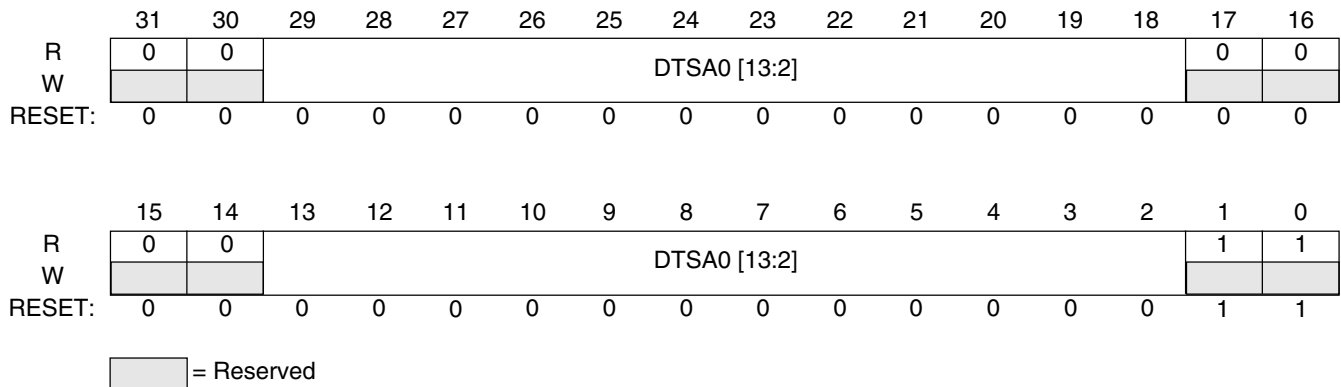


Figure 60-19. Data Trace Address Range 0 Register (NDEDI_DTAR0)

Table 60-20. NDEDI_DTAR0 field descriptions

Field	Description
DTSA0	Data Trace Start Address 0 The DTSA0 field is the start address for ETPU data trace window 0.
DTEA0	Data Trace End Address 0 The DTEA0 field is the end address for ETPU data trace window 0.

2. If RC0 equals 1 for a determined source all accesses from that source will have the address value acknowledged.

60.6.1.17 Data Trace Address Range 1 Register (NDEDI_DTAR1)

The NDEDI_DTAR1 register defines a data trace address range. The address range is shared by the two ENGINES and the CDC. All three of these clients can independently enable the range for reads and/or writes. If the start address value is greater than the end address value, all accesses are considered to have the address value outside to the data trace window 1.³

Notice that the combination of the start and end addresses for data trace into one register does not follow the standard's recommended method of having separate registers for start and end addresses.

Client Select: ENGINE1_SRC or ENGINE2_SRC or CDC_SRC

Register index: 66

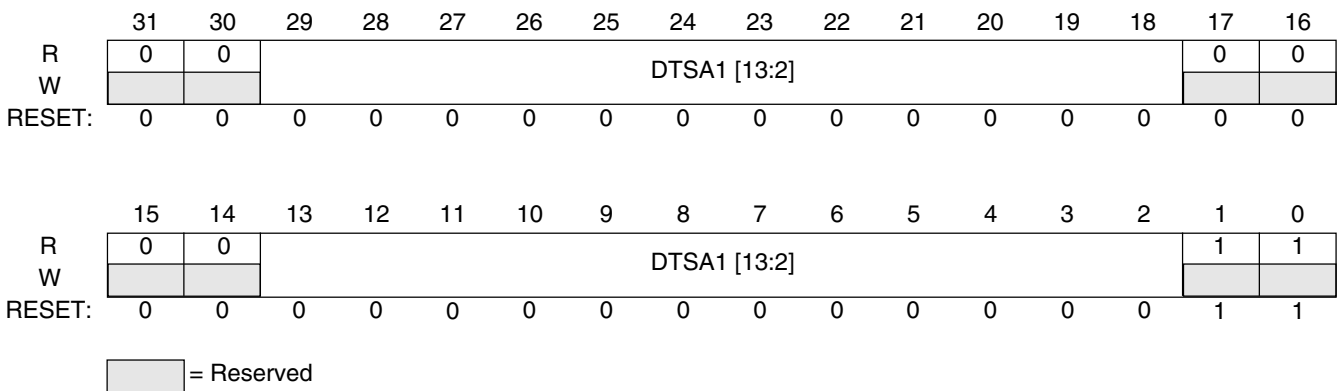


Figure 60-20. Data Trace Address Range 1 Register (NDEDI_DTAR1)

Table 60-21. NDEDI_DTAR1 field descriptions

Field	Description
DTSA1	Data Trace Start Address 1 The DTSA1 field is the start address for ETPU data trace window 1.
DTEA1	Data Trace End Address 1 The DTEA1 field is the end address for ETPU data trace window 1.

3. If RC1 equals 1 for a determined source all accesses from that source will have the address value acknowledged.

60.6.1.18 Data Trace Address Range 2 Register (NDEDI_DTAR2)

The NDEDI_DTAR2 register defines a data trace address range. The address range is shared by the two ENGINES and the CDC. All three of these clients can independently enable the range for reads and/or writes. If the start address value is greater than the end address value, all accesses are considered to have the address value outside to the data trace window 2.⁴

Notice that the combination of the start and end addresses for data trace into one register does not follow the standard's recommended method of having separate registers for start and end addresses.

Client Select: ENGINE1_SRC or ENGINE2_SRC or CDC_SRC

Register index: 67

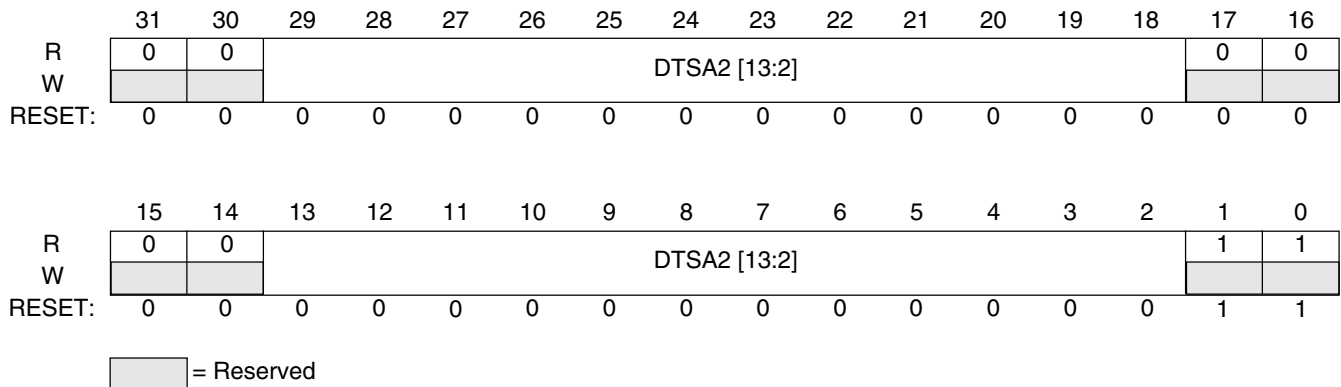


Figure 60-21. Data Trace Address Range 2 Register (NDEDI_DTAR2)

Table 60-22. NDEDI_DTAR2 field descriptions

Field	Description
DTSA2	Data Trace Start Address 2 The DTSA2 field is the start address for ETPU data trace window 2.
DTEA2	Data Trace End Address 2 The DTEA2 field is the end address for ETPU data trace window 2.

4. If RC2 equals 1 for a determined source all accesses from that source will have the address value acknowledged.

60.6.1.19 Data Trace Address Range 3 Register (NDEDI_DTAR3)

The NDEDI_DTAR3 register defines a data trace address range. The address range is shared by the two ENGINES and the CDC. All three of these clients can independently enable the range for reads and/or writes. If the start address value is greater than the end address value, all accesses are considered to have the address value outside to the data trace window 3.⁵

Notice that the combination of the start and end addresses for data trace into one register does not follow the standard's recommended method of having separate registers for start and end addresses.

Client Select: ENGINE1_SRC or ENGINE2_SRC or CDC_SRC

Register index: 68

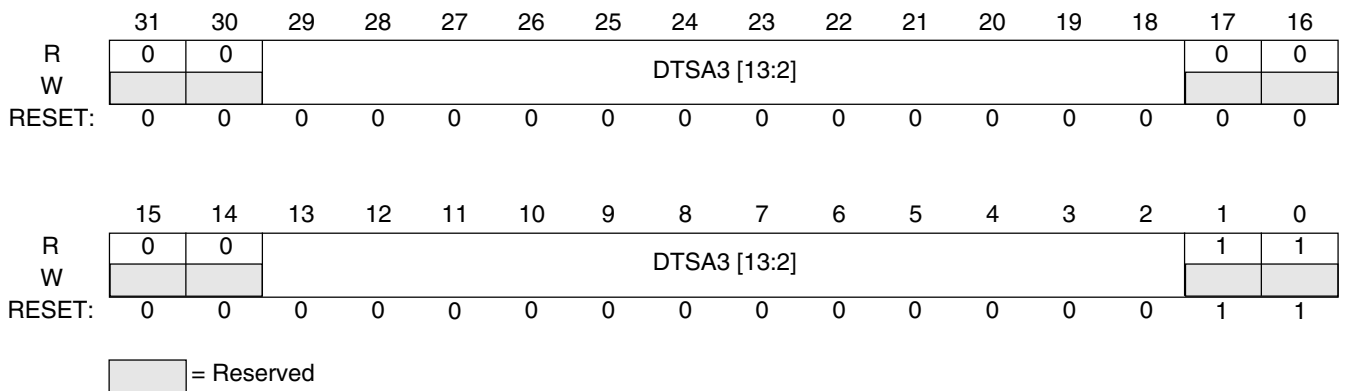


Figure 60-22. Data Trace Address Range 3 Register (NDEDI_DTAR3)

Table 60-23. NDEDI_DTAR3 field descriptions

Field	Description
DTSA3	ETPU Data Trace Start Address 3 The DTSA3 field is the start address for ETPU data trace window 3.
DTEA3	ETPU Data Trace End Address 3 The DTEA3 field is the end address for ETPU data trace window 3.

60.6.2 Unimplemented Registers

Unimplemented registers are those with client select and index value combinations other than those listed in Table 60-3 . NDEDI will treat unimplemented registers like the JTAG BYPASS instruction.

5. If RC3 equals 1 for a determined source all accesses from that source will have the address value acknowledged.

60.7 Functional description

This section describes the functional description.

60.7.1 NDEDI reset configuration

This section describes the reset configuration of NDEDI.

60.7.1.1 Enabling NDEDI Class 1 operation

The NDEDI Class 1 features are always enabled after exiting the JTAG Test-Logic-Reset state. But they are disabled while the JTAG is in this state. Thus, allowing low power mode for a production part. The registers are always reset while in the JTAG Test-Logic-Reset state.

60.7.1.2 Enabling NDEDI Class 3 operation

When NDEDI input `nex_mcko_en` is asserted the NDEDI Classes 1 and 3 are enabled after exiting the JTAG Test-Logic-Reset state, otherwise the NDEDI Class 3 features will be disabled, entering in the Disable-Port Mode, thus no trace output will be provided, and auxiliary port output pins will be disabled (driven inactive or used for an alternate function if sharing pins).⁶

60.7.2 Auxiliary output port

This section describes the Auxiliary output ports.

60.7.2.1 Output Message Protocol

The protocol for transmitting messages via the auxiliary port is accomplished with the $\overline{\text{MSEO}}$ functions. $\overline{\text{MDO}}$ and $\overline{\text{MSEO}}$ must be sampled by the development tool on the rising edge of MCKO .

6. Class 1 features are still available in Disable-Port Mode.

The following figure illustrates the state diagram for $\overline{\text{MSEO}}$ transfers. All transitions not included in the figure are reserved, and are never generated by the NDEDI block.

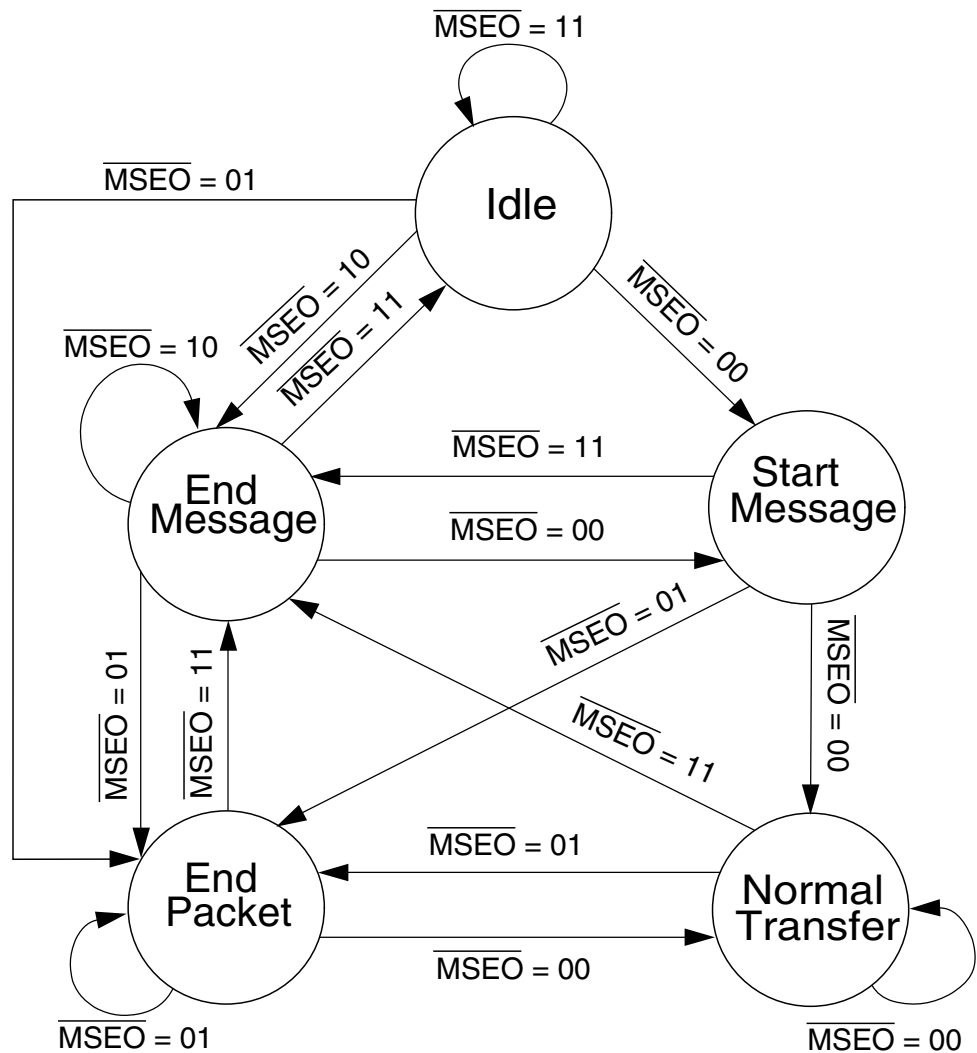


Figure 60-23. $\overline{\text{MSEO}}$ Transfers

60.7.2.2 Output messages

The following table describes the messages that the NDEDI can transmit on the auxiliary port.

Table 60-24. NDEDI messages

Message Name	Min. Packet Size (bits)	Max Packet Size (bits)	Packet Type	Packet Name	Packet Description
Debug Status Message	6	6	fixed	TCODE	Value = 0
	K	K	fixed	SRC	Client that is the source of the message
	32	32	fixed	STATUS	Value of the Development Status register
Ownership Trace Message	6	6	fixed	TCODE	Value = 2
	K	K	fixed	SRC	Client that is the source of the message
	16	16	fixed	PROCESS	Process ID. This value depends on the ETPU state at the beginning of a service
Data Trace, Data Write Message	6	6	fixed	TCODE	Value = 5
	K	K	fixed	SRC	Client that is the source of the message
	2	2	fixed	SIZE	Size of data access (8, 24, or 32 bits)
	1	14	variable	U-ADDR	Unique portion of the data write address. Most significant bits that have 0 values may be truncated
	1	32	variable	DATA	Data value written
Data Trace, Data Read Message	6	6	fixed	TCODE	Value = 6
	K	K	fixed	SRC	Client that is the source of the message
	2	2	fixed	SIZE	Size of data access (8, 24, or 32 bits)
	1	14	variable	U-ADDR	Unique portion of the data read address. Most significant bits that have 0 values may be truncated
	1	32	variable	DATA	Data value read
Error Message	6	6	fixed	TCODE	Value = 8
	K	K	fixed	SRC	Client that is the source of the message
	5	5	fixed	ECODE	Error Code
Data Trace, Data Write with Sync Message	6	6	fixed	TCODE	Value = 13
	K	K	fixed	SRC	Client that is the source of the message
	2	2	fixed	SIZE	Size of data access (8, 24, or 32 bits)
	1	14	variable	F-ADDR	Full address of the memory location written. Most significant bits that have 0 values may be truncated
	1	32	variable	DATA	Data value written
Data Trace, Data Read with Sync Message	6	6	fixed	TCODE	Value = 14
	K	K	fixed	SRC	Client that is the source of the message
	2	2	fixed	SIZE	Size of data access (8, 24, or 32 bits)
	1	14	variable	F-ADDR	Full address of the memory location read. Most significant bits that have 0 values may be truncated
	1	32	variable	DATA	Data value read
Watchpoint Hit Message	6	6	fixed	TCODE	Value = 15
	K	K	fixed	SRC	Client that is the source of the message
	7	7	fixed	WPHIT	Number indicating watchpoint source

Table continues on the next page...

Table 60-24. NDEDI messages (continued)

Message Name	Min. Packet Size (bits)	Max Packet Size (bits)	Packet Type	Packet Name	Packet Description
Resource Full Message	6	6	fixed	TCODE	Value = 27
	K	K	fixed	SRC	Client that is the source of the message
	4	4	fixed	RCODE	Resource Code. Refer to Table 60-26
	1	H	variable	HIST	Branch/Predicate Instruction History. This packet is terminated by a stop bit set to 1 after the last history bit. This allows the tool to determine which bits are part of the history packet and which are padded zeros
Program Trace, Indirect Branch with History Message	6	6	fixed	TCODE	Value = 28
	K	K	fixed	SRC	Client that is the source of the message
	1	8	variable	I-CNT	Number of instruction units executed since the last taken branch, not taken direct branch, or predicated instruction
	1	14	variable	U-ADDR	Unique portion of the branch target address for an indirect branch. Most significant bits that have 0 values may be truncated
	1	H	variable	HIST	Branch/Predicate Instruction History. This packet is terminated by a stop bit set to 1 after the last history bit. This allows the tool to determine which bits are part of the history packet and which are padded zeros
Program Trace, Indirect Branch with History Sync Message	6	6	fixed	TCODE	Value = 29
	K	K	fixed	SRC	Client that is the source of the message
	1	8	variable	I-CNT	Number of instruction units executed since the last taken branch, not taken direct branch, or predicated instruction
	1	14	variable	F-ADDR	Full branch target address for an indirect branch. Most significant bits that have 0 values may be truncated
	1	H	variable	HIST	Branch/Predicate Instruction History. This packet is terminated by a stop bit set to 1 after the last history bit. This allows the tool to determine which bits are part of the history packet and which are padded zeros
Program Trace Correlation Message	6	6	fixed	TCODE	Value = 33
	K	K	fixed	SRC	Client that is the source of the message
	4	4	fixed	EV-CODE	Event Code
	1	8	variable	I-CNT	Number of instruction units executed since the last taken branch, not taken direct branch, or predicated instruction
	1	H	variable	HIST	Branch/Predicate Instruction History. This packet is terminated by a stop bit set to 1 after the last history bit. This allows the tool to determine which bits are part of the history packet and which are padded zeros

Table continues on the next page...

Table 60-24. NDEDI messages (continued)

Message Name	Min. Packet Size (bits)	Max Packet Size (bits)	Packet Type	Packet Name	Packet Description
Program Trace, Channel Start Service Message	6	6	fixed	TCODE	Value = 58 (Vendor-Defined Message)
	K	K	fixed	SRC	Client that is the source of the message
	5	5	fixed	S-CHAN	Number of the channel being serviced
	1	8	variable	I-CNT	Number of instruction units executed since the last taken branch, not taken direct branch, or predicated instruction
	1	14	variable	F-ADDR	Full address of the first instruction. Most significant bits that have 0 values may be truncated.
	1	H	variable	HIST	Branch/Predicate Instruction History. This packet is terminated by a stop bit set to 1 after the last history bit. This allows the tool to determine which bits are part of the history packet and which are padded zeros
Program Trace, Channel Trace Enable Message	6	6	fixed	TCODE	Value = 59 (Vendor-Defined Message)
	K	K	fixed	SRC	Client that is the source of the message
	5	5	fixed	S-CHAN	Number of the channel being serviced
	1	14	variable	F-ADDR	Full address of the first instruction after the program trace is enabled by a watchpoint. Or, the full address of the first instruction after the Event Queue gets empty after a Queue Overrun. Most significant bits that have 0 values may be truncated
	0	5	variable	CHAN	Channel Register Value. Only sent if different from S-CHAN value
Channel Register Write Message	6	6	fixed	TCODE	Value = 60 (Vendor-Defined Message)
	K	K	fixed	SRC	Client that is the source of the message
	1	5	variable	CHAN	Value written to the channel register

The following table describes the Error Code encodings for the error messages generated by the NDEDI block.

Table 60-25. Error Codes Encodings(ECODE)

Error Code	Description
0b00000	Ownership trace overrun
0b00001	Program trace overrun
0b00010	Data trace overrun
0b00110	Watchpoint Overrun
0b00111	Program and/or Data and/or Ownership Trace Overrun
0b01000	Program Trace and/or Data Trace and/or Ownership Trace and/or Watchpoint Overrun
0b11000	Debug Status Overrun

Table continues on the next page...

Table 60-25. Error Codes Encodings(ECODE) (continued)

Error Code	Description
0b11001	Debug Status Overrun and/or Program and/or Data and/or Ownership Trace Overrun
0b11010	Debug Status and/or Program Trace and/or Data Trace and/or Ownership Trace and/or Watchpoint Overrun

The following table describes the Resource Code encodings for the resource full messages.

Table 60-26. Resource Codes Encodings

Resource Code	Resource	DATA Packet Count	DATA Packet Value
0b0000	Program Trace, Sequential Counter	1	Branch/Predicate Instruction History. This packet is terminated by a stop bit set to 1 after the last history bit. The value that causes the counter to fill is always the same and can be determined without sending the value
0b0001	Program Trace, Branch/Predicate History	1	Branch/Predicate Instruction History. This packet is terminated by a stop bit set to 1 after the last history bit

Message formatting is performed by the message formatter block. Raw messages read from the message queues are independent of the number of MDO pins implemented.

The following table shows the various message formats and the location of variable length packets. Notice that for variable-length packets, the transmitted size of the packet is determined from the range of the least significant bit to the most significant non-zero-valued bit (i.e. most significant 0 value bits are not transmitted).

Table 60-27. Message Packet Sizes

Message	TCODE	Packet #1	Packet #2	Packet #3	Packet #4	Packet #5	Packet #6	Min. Size ¹ (bits)	Max Size ² (bits)
Debug Status Message	0	Fixed = K	Fixed =32	NA	NA	NA	NA	38+K	38+K
Ownership Trace Message	2	Fixed = K	Fixed =16	NA	NA	NA	NA	22+K	22+K
Data Trace, Data Write Message	5	Fixed = K	Fixed ³ = 2	Variable Min = 1 Max = 14	Variable Min = 1 Max = 32	NA	NA	10+K	54+K
Data Trace, Data Read Message	6	Fixed = K	Fixed ³ = 2	Variable Min = 1 Max = 14	Variable Min = 1 Max = 32	NA	NA	10+K	54+K

Table continues on the next page...

Table 60-27. Message Packet Sizes (continued)

Message	TCODE	Packet #1	Packet #2	Packet #3	Packet #4	Packet #5	Packet #6	Min. Size ¹ (bits)	Max Size ² (bits)
Error Message	8	Fixed = K	Fixed = 5	NA	NA	NA	NA	11+K	11+K
Data Trace, Data Write with Sync Message	13	Fixed = K	Fixed ³ = 2	Variable Min = 1 Max = 14	Variable Min = 1 Max = 32 q	NA	NA	10+K	54+K
Data Trace, Data Read with Sync Message	14	Fixed = K	Fixed ³ = 2	Variable Min = 1 Max = 14	Variable Min = 1 Max = 32	NA	NA	10+K	54+K
Watchpoint Hit Message	15	Fixed = K	Fixed = 7	NA	NA	NA	NA	13+K	13+K
Resource Full Message	27	Fixed = K	Fixed = 4	Variable Min = 1 Max = H	NA	NA	NA	11+K	10+H+K
Program Trace, Indirect Branch w/ History Message	28	Fixed = K	Variable Min = 1 Max = 8	Variable Min = 1 Max = 14	Variable Min = 1 Max = H	NA	NA	9+K	28+H+K
Program Trace, Indirect Branch w/ History Sync Message	29	Fixed = K	Variable Min = 1 Max = 8	Variable Min = 1 Max = 14	Variable Min = 1 Max = H	N/A	N/A	9+K	28+H+K
Program Trace, Correlation Message	33	Fixed = K	Fixed=4	Variable Min = 1 Max = 8	Variable Min = 1 Max = H	NA	NA	12+K	18+H+K
Program Trace, Channel Start Service Message	58	Fixed = K	Fixed = 5	Variable Min = 1 Max = 8	Variable Min = 1 Max = 14	Variable Min = 1 Max = H	NA	14+K	33+H+K
Program Trace, Channel Trace Enable Message	59	Fixed = K	Fixed=5	Variable Min = 1 Max = 14	Variable Min = 0 Max = 5	NA	NA	12+K	30+K

Table continues on the next page...

Table 60-27. Message Packet Sizes (continued)

Message	TCODE	Packet #1	Packet #2	Packet #3	Packet #4	Packet #5	Packet #6	Min. Size ¹ (bits)	Max Size ² (bits)
Channel Register Write Message	60	Fixed = K	Variable Min = 1 Max = 5	NA	NA	NA	NA	7+K	11+K

1. Minimum information size. The actual number of bits transmitted depends on the number of MDO pins.
2. Maximum information size. The actual number of bits transmitted depends on the number of MDO pins.
3. Packet is fixed and NDEDI always transmits 2 bits, but other sources sharing the auxiliary port may have a different packet size{ndedi_msg_checker, ndedi_dt_rnd}. Packet size can always be determined by the value of the SRC packet.
{Statement}

The double edges in the preceding table indicate the starts and ends of messages. The shaded edges in the preceding figure indicate the end of a variable length packet which is not also the end of the message. Packets without shaded areas between them are grouped into super packets and are transmitted together without end-of-packet indications between them.

60.7.2.2.1 Rules of messaging

- A variable-length packet within a message must end on a port boundary (Port boundaries depend on the number of MDO pins active with the current reset configuration)
- A variable-length packet may start within a port boundary only when following a fixed-length packet.
- Superblocks must end on a port boundary
- When a variable-length packet is sized such that it does not end on a port boundary, it is necessary to extend and zero fill the remaining bits after the highest order bit so that it can end on a port boundary
- Multiple fixed-length packets may start and/or end on the same clock
- When any packet follows a variable-length packet, it must start on a port boundary
- The packet containing the TCODE number is always transferred out first, followed by subsequent packets of information
- Within a packet, the least significant bits are shifted out first. The following figure shows the transmission sequence of a message that is made up of a TCODE followed by two packets

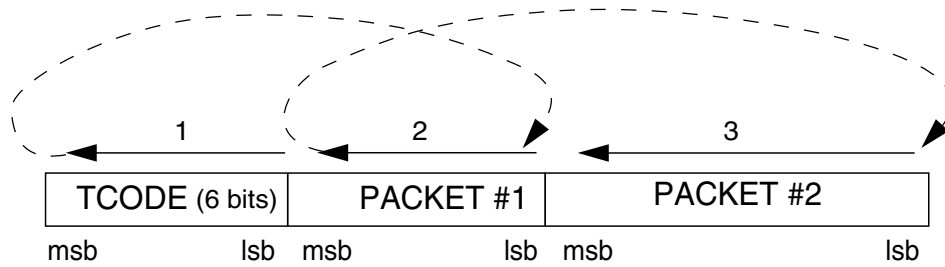


Figure 60-24. Transmission Sequence of Messages

60.7.2.2.2 Examples

The following are examples of branch trace and data trace messages.

The following figure illustrates how the indirect branch with history public message is transmitted. *Notice that the example uses a 4-bit MDO port*. For this example, 4 bits are shifted out in the I-CNT packet, 5 bits are shifted out in the HIST packet, and 11 bits are shifted out for the UADDR packet. Notice also that T0, S0, H0, I0, and A0 are the least significant bits where:

Tx= TCODE packet bits (Fixed)

Sx= SRC packet bits (Fixed)

Hx = HIST packet bits (Variable)

Ix= I-CNT packet bits (Variable)

Ax= UADDR packet bits (Variable)

Clock	MDO[3:0]				MSEO[1:0]	State
	3	2	1	0		
0	X	X	X	X	X	End Message or Idle
1	T3	T2	T1	T0	00	Start Message
2	S1	S0	T5	T4	00	Normal Transfer
3	I2	I1	I0	S2	00	Normal Transfer
4	0	0	0	I3	01	End Packet
5	A3	A2	A1	A0	00	Normal Transfer
6	A7	A6	A5	A4	00	Normal Transfer
7	0	A10	A9	A8	01	End Packet
8	H3	H2	H1	H0	00	Normal Transfer
9	0	0	0	H4=1	11	End Message

Figure 60-25. Indirect Branch with History Message (4 MDO pins)

The following figure shows the same message being sent using 8 MDO pins

Clock	MDO[7:0]								$\overline{\text{MSEO}}[1:0]$	State
	7	6	5	4	3	2	1	0		
0	X	X	X	X	X	X	X	X	X	End Message or Idle
1	S1	S0	T5	T4	T3	T2	T1	T0	00	Start Message
2	0	0	0	I3	I2	I1	I0	S2	01	End Packet
4	A7	A6	A5	A4	A3	A2	A1	A0	00	Normal Transfer
5	0	0	0	0	0	A10	A9	A8	01	End Packet
3	0	0	0	H4=1	H3	H2	H1	H0	11	End Message

Figure 60-26. Indirect Branch with History Message (8 pins)

The following figure illustrates how a data write with sync message is transmitted. *Notice that the example uses a 16-bit MDO port*. For this example, 7 bits are sent for the F-ADDR packet, and 5 bits are sent for the DATA packet. Notice also that T0, S0, A0, and D0 are the least significant bits where:

Tx= TCODE packet bits (Fixed)

Sx= SRC packet bits (Fixed)

Zx = SIZE packet bits (Fixed)

Dx= DATA packet bits (Variable)

Ax= F-ADDR packet bits (Variable)

Clock	MDO[15:0]																$\overline{\text{MSEO}}[1:0]$	State
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0		
0	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	End Message or Idle
1	A4	A3	A2	A1	A0	Z1	Z0	S2	S1	S0	T5	T4	T3	T2	T1	T0	00	Start Message
2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	A6	A5	01	End Packet
3	0	0	0	0	0	0	0	0	0	0	0	D4	D3	D2	D1	D0	11	End Message

Figure 60-27. Data Write with Sync Message (16 pins)

60.7.2.2.3 Temporal Ordering of Transmitted Messages

All Messages are sent out in the sequence their related event actually occurred. The ETPU interface with the NDEDI provides the ETPU signals to be monitored and captured. At the occurrence of any event that would cause a message, a snapshot of all information needed by the Message Formatter to generate a message is queued. Doing so, several events from either Engines or CDC can be queued at the same time.

Whenever the auxiliary port is idle, the Event Queue is read and the appropriate messages are formatted. Then, NDEDI sends a request for the auxiliary port, and as soon as the auxiliary port is granted the message is sent. If more than one messages are to be sent, the Message Formatter respects the following priority: Messages related to Error, Debug Status, Watchpoints, Program Trace and Data Trace. A request for the auxiliary port is sent for every messages within the snapshot.

NDEDI maintain temporal ordering of messages relative to ENGINE1, ENGINE2 and CDC, but not relative to messages generated by other clients sharing the auxiliary port.

The Time Slot Transition (TST) is considered to be an active state, and since data are accessed at the SPRAM, it is possible that Data Trace Events are recognized within TST.

⁷ For further information about TST, please refer to the ETPU BG.

60.7.3 IEEE 1149.1 (JTAG) Input Port

The NDEDI block uses the IEEE 1149.1 TAP for accessing registers. This port is shared among all TAP controllers on the MCU. This sharing is controlled by the value in the instruction register and client select control register.

The NDEDI implements a Nexus controller state machine that transitions based on the state of the IEEE 1149.1 state machine shown in [Figure 60-29](#) . The Nexus controller state machine, shown in [Figure 60-30](#), is defined by the IEEE-ISTO 5001-2002 standard.

The instructions implemented by the NDEDI TAP controller are listed in the following table. If any other instruction is received it will be treated as a BYPASS.

7. Due to ETPU operation, speculative accesses may also happen within TST and will be traced.

Table 60-28. Implemented Instructions

Instruction Name	Priv ate/ Pub lic	Opcode	Description
NEXUS-ENABLE	publi c	4'b0000	Activate Nexus controller state machine to read and write NDEDI registers
BYPASS	publi c	4'b1111	Implements a single shift register stage providing a minimum serial length between TDI and TDO

Data is shifted between TDI and TDO starting with the least significant bit, as illustrated in the following figure. This applies for the instruction register and all Nexus tool-mapped registers.

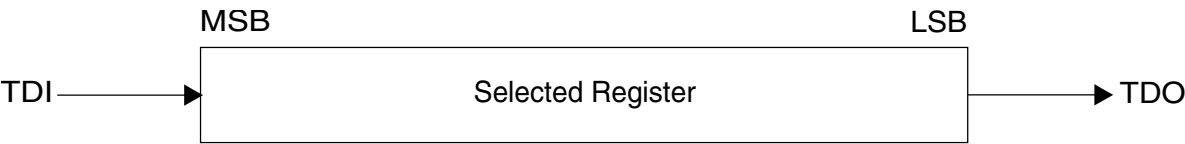


Figure 60-28. Shifting Data Into Register

60.7.3.1 Enabling the NDEDI TAP Controller

Assertion of the $\overline{\text{TRST}}$ input resets all TAP controllers on the MCU. The controllers are loaded with the BYPASS instruction upon exit of the 'TEST-LOGIC-RESET' JTAG controller state. Loading the NEXUS-ENABLE instruction grants access to Nexus debug. NDEDI acts as BYPASS by loading any other instruction. The TDO bit is driven with the value of the instruction register while in 'SHIFT_IR' state.

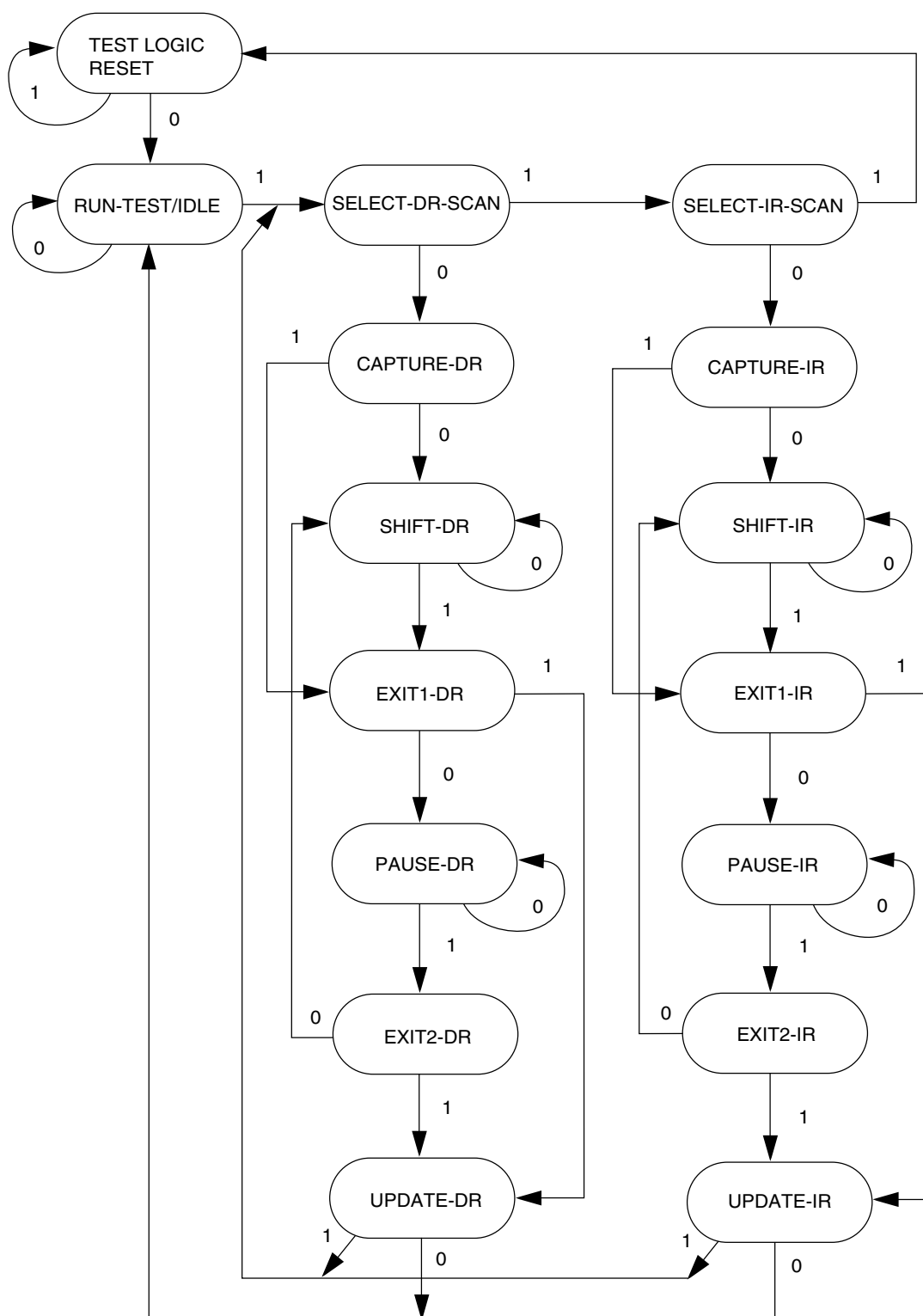


Figure 60-29. IEEE 1149.1 16-State Finite State Machine

60.7.3.2 Loading NEXUS-ENABLE Instruction

Access to the NDEDI registers is enabled when the TAP controller instruction register is loaded with the NEXUS-ENABLE instruction. The current instruction value is loaded into the IEEE 1149.1 shifter in the 'CAPTURE_IR' state. The instruction is shifted in via the 'SELECT-IR-SCAN' path and loaded in the 'UPDATE-IR' state. At this point, the Nexus controller state machine, shown in the following figure, transitions to the 'REG_SELECT' state. The following table illustrates the IEEE 1149.1 sequence to load the NEXUS-ENABLE instruction.

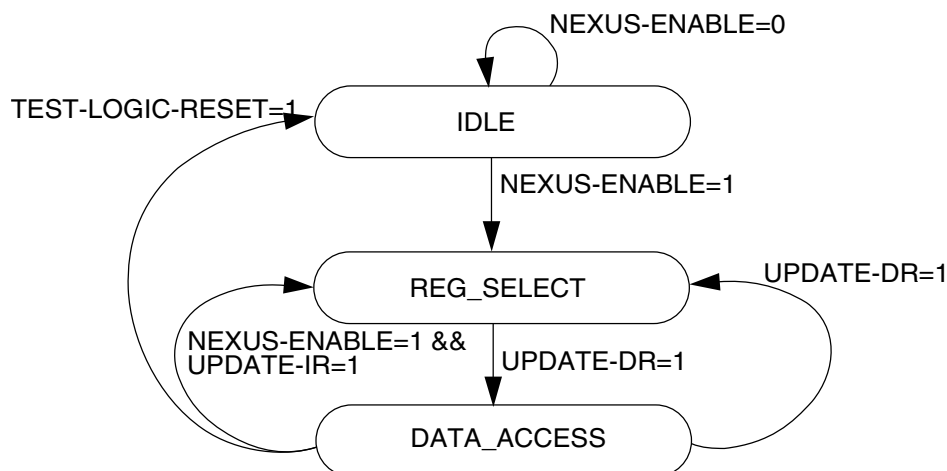


Figure 60-30. Nexus Controller State Machine

Table 60-29. Loading NEXUS-ENABLE instruction

Clock	TM S	IEEE 1149.1 State	Nexus State	Description
0	0	RUN-TEST/IDLE	IDLE	IEEE 1149.1 controller in idle state
1	1	SELECT-DR-SCAN	IDLE	Transitional state
2	1	SELECT-IR-SCAN	IDLE	Transitional state
3	0	CAPTURE-IR	IDLE	Internal shifter loaded with current instruction
4	0	SHIFT-IR	IDLE	TDO becomes active, and the IEEE 1149.1 shifter is ready. Shift in all but the last bit of the NEXUS_ENABLE instruction
4 TCKs				
12	1	EXIT1-IR	IDLE	Last bit of instruction shifted in
13	1	UPDATE-IR	IDLE	NEXUS-ENABLE loaded into instruction register
14	0	RUN-TEST/IDLE	REG_SELECT	NEXUS-ENABLE ready to be read

60.7.3.3 Selecting an IEEE 1149.1 Register

When the NEXUS-ENABLE instruction is loaded in the NDEDI TAP controller instruction register, the input port allows development tool access to all NDEDI registers. Each register has a 7-bit address index. Some clients have registers with the same index as registers in other clients. Therefore, to access the registers of a particular client, the Client Select Control register must be set for that client.

All register access is performed via the 'SELECT-DR-SCAN' path. The Nexus state machine defaults to the register select state when enabled. Accessing a register requires two passes through the 'SELECT-DR-SCAN' path: one pass to select the register and the second pass to read/write the register.

The first pass through the 'SELECT-DR-SCAN' path is used to enter an 8-bit Nexus command consisting of a read/write control bit in the LSB followed by a 7-bit register address index, as illustrated in the following figure. The read/write control bit is set to 1 for writes and 0 for reads. The current value of the Controller Command Input is captured into the IEEE 1149.1 shifter during the 'CAPTURE-DR' state while the Nexus controller state machine is in the 'REG_SELECT' state.

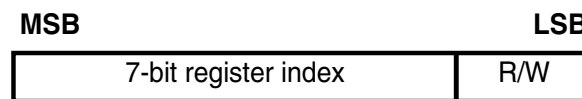


Table 60-30. Writing to a Register (continued)

Clock	TMS	IEEE 1149.1 State	Nexus State	Description
3	0	SHIFT-DR	REG_SELECT	TDO becomes active, and write bit and 6 bits of register index shifted in
7 TCKs				
12	1	EXIT1-DR	REG_SELECT	Last bit of register index shifted into TDI
13	1	UPDATE-DR	REG_SELECT	Controller decodes and selects register
14	1	SELECT-DR-SCAN	DATA_ACCESS	Second pass through 'SELECT-DR-SCAN' path
15	0	CAPTURE-DR	DATA_ACCESS	Internal shifter loaded with current value of register
16	0	SHIFT-DR	DATA_ACCESS	TDO becomes active, and outputs current value of register while new value is shifted in through TDI
31 TCKs				
48	1	EXIT1-DR	DATA_ACCESS	Last bit of current value shifted out TDO. Last bit of new value shifted in TDI
49	1	UPDATE-DR	DATA_ACCESS	Value written to register
50	0	RUN-TEST/IDLE	REG_SELECT	Controller returned to idle state. It could also return to 'SELECT-DR-SCAN' to write another register

60.7.4 Microcode Development Support

The NDEDI contains a number of hardware hooks that aid in the development of microcode for the ETPUs. This features described in this section make the ETPUs compliant with Nexus Class 1 features.

The main features provided are the abilities to:

- Read and write ETPU internal registers in debug mode
- Read and write SPRAM in debug mode
- Enter a debug mode at reset negation
- Enter a debug mode during normal execution
- Single step instructions and re-enter debug mode
- Stop program execution on instruction/data breakpoints or channel service breakpoints and enter debug mode
- Set breakpoints or watchpoints

- Execute external microcode instruction in debug mode
- Read/write the microprogram counter register value in debug mode

Refer to [Initialization/Application Information](#) on how the following functions provide the features listed above.

60.7.5 Debug Status

The NDEDI module provides the Debug Status via the auxiliary port, as defined by the IEEE-ISTO 5001-2002 standard.

60.7.5.1 Messaging

The NDEDI block provides debug status messaging using IEEE-ISTO 5001-2002 standard-defined public messages. When the development status register changes, a debug status change event is sent to the Event Queue. If the debug status change condition occurs while the Event Queue is not enabled for storing snapshots, a debug status overrun message is generated.

The debug status message has the format shown in the following figure.

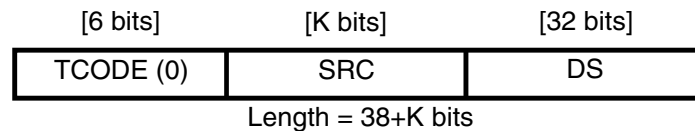


Figure 60-32. Debug Status Message Format

The DS packet correspond to the ENGINE_x DS register value described in [Figure 60-4](#).

60.7.5.2 Error Messages

A debug status overrun error event is queued if a debug status event occurs while the Event Queue is not enabled for storing snapshots. The error event is stored as soon as the Event Queue is enabled which happens when the queue gets empty.

The error message has the format shown in the following figure.

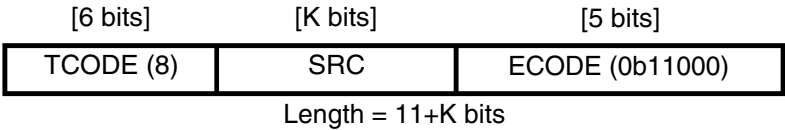


Figure 60-33. Debug Status Error Message Format

60.7.5.3 Synchronization

Upon the exit of debug mode, the next program and data trace messages are synchronization messages. The exit of debug mode by one of the Engines causes data and program synchronization for that Engine only.

60.7.5.4 Timing Diagrams

Note that all of the following timing diagrams assume a 3-bit SRC size and a 16-bit MDO port. The state variable is not a signal, but instead is derived from $\overline{\text{MSEO}}$. It is included for clarity. Refer to [Figure 60-23](#) for $\overline{\text{MSEO}}$ state diagram.

The following abbreviations are used for the state variable in the diagrams:

- ID = Idle
- SM = Start Message
- NT = Normal Transfer
- EP = End Packet
- EM = End Message

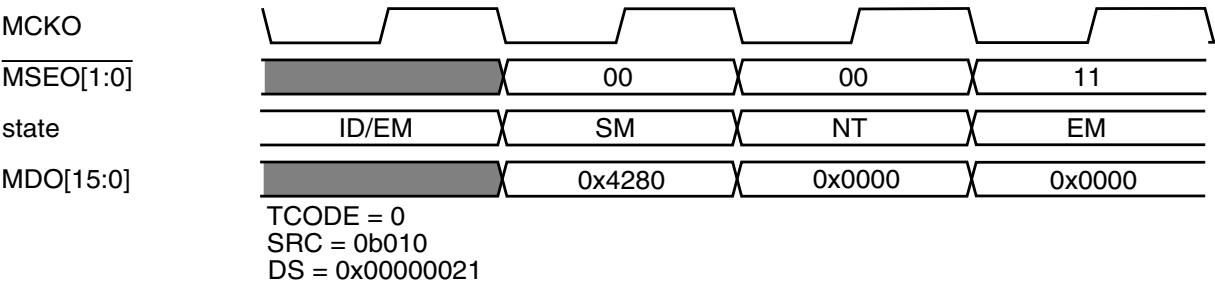


Figure 60-34. Debug Status Message

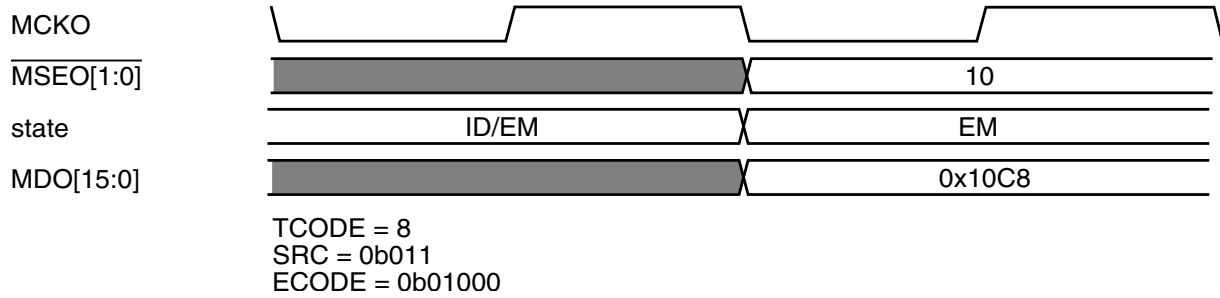


Figure 60-35. Debug Status Overrun Error Message

60.7.6 Ownership Trace

Ownership trace provides a macroscopic view of the ETPU program flow. This is done by generating an ownership trace event at the start of each channel service and whenever an engine becomes idle.

Each engine within the ETPU system generates ownership traces independently, and the events are queued in the order in which they occur.

60.7.6.1 Messaging

Ownership trace information is transmitted via the auxiliary port using an ownership trace message (OTM).

Ownership trace information is transmitted in the format shown in the following figure.

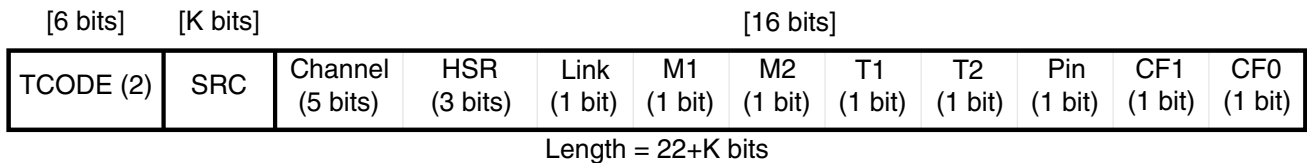


Figure 60-36. Ownership Trace Message Format

Notice that the channel number and entry points number in the preceding figure are concatenated to form the variable length PROCESS packet defined by the IEEE-ISTO 5001-2002 standard.

60.7.6.2 OTM Flow

A channel generates an ownership trace message when service starts if ownership trace is enabled for the ETPU via the development control register. If ownership trace is enabled and an engine becomes idle, NDEDI will transmit a special OTM with the PROCESS packet set to 0.

60.7.6.3 Timing Diagram

The following figure shows an example of an ownership message being transmitted on the auxiliary port. The diagram assumes a 3-bit SRC size and a 16-bit MDO port. The state variable is not a signal but instead, it is derived from $\overline{\text{MSEO}}$. It is included for clarity. Refer to [Figure 60-23](#) for $\overline{\text{MSEO}}$ state diagram.

The following abbreviations are used for the state variable in the diagrams:

- ID = Idle
- SM = Start Message
- NT = Normal Transfer
- EP = End Packet
- EM = End Message

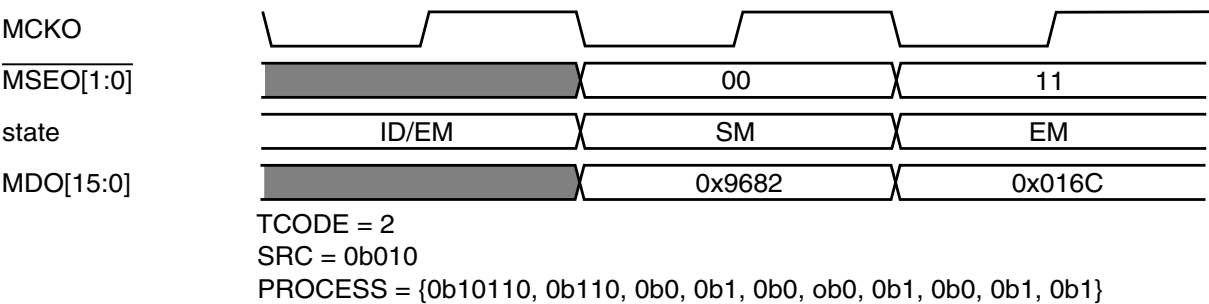


Figure 60-37. Ownership Trace Message

60.7.7 Program Trace

This section details the program trace mechanism supported by the NDEDI for the dual Engine system. Program trace is implemented via a combination of Branch Trace Messaging (BTM) and Ownership Trace Messaging (OTM) as per the IEEE-ISTO 5001-2002 standard definition.

60.7.7.1 Branch Trace Messaging

Branch trace messaging facilitates program trace by providing the following types of information:

- Messages for the start of channel services, each containing the full address of the first instruction to execute and history and sequential count information left over from the previously traced channel service. This message is always synchronizing
- Messages to indicate that the Program Trace was enabled in the middle of a thread or after the Queue is enabled for storing snapshots and it was previously disabled due to queue overrun. It includes the address of the enable point, the value of the serviced channel and the CHAN reg value only if it differs from the serviced channel value
- Messages to indicate that the internal history buffer has filled and been reset. The current value of the history buffer is included in each of these messages
- Messages to indicate that the internal sequential instruction counter has overflow and been reset. The current value of the history buffer is included in each of these messages
- Messages for taken indirect branches, each including a branch/predicate instruction⁸ history packet, sequential instruction count, and the unique portion of the branch target address. Indirect branch messages that are generated with pending synchronization event include the full branch target address (instead of the unique portion of the address)
- Messages to indicate the entry into debug mode, the entry into low power mode, or the end of channel service with no new service pending
- Messages for writes to the CHAN register, indicating the value assigned to the register

The address packet of a program trace address excludes the 2 least significant bits of the byte address because they are zero for all instruction addresses.

60.7.7.2 Branch Trace Message Formats

There are seven types of messages used by the ETPU for branch tracing. These types are shown in the following table.

8. Predicate Instructions depend on the AS/CE field within ETPU instructions.

Table 60-31. ETPU BTM Messages

Message	Synchronizing	Public/Private
Resource Full Message	No	Public
Indirect Branch with History Message	No	Public
Indirect Branch with History Synchronization Message	Yes	Public
Program Trace Correlation Message	No	Public
Channel Start Service Message	Yes	Private
Channel Trace Enable Message	Yes	Private
Channel Register Write Message	No	Private

The occurrence of any of the following conditions requires subsequent synchronization:

- Exit of system reset
- Exit of low power mode
- Exit of debug mode
- Write to the Development Control Register or the Program Trace Channel Enable Register⁹
- 255 branch trace messages of the same are queued without synchronization
- Assertion of the Event In ($\overline{\text{EVTI}}$) pin, depending on how the development control register is configured
- A watchpoint occurrence¹⁰
- A queue overrun

The message that follows any of these synchronizing events will be a synchronizing message if it is not a Resource Full Message or a Channel Register Write Message which have no synchronizing information. If the ETPU reaches the idle the next program trace message will be a Channel Start Service Message or a Channel Trace Enable Message, that are synchronizing.

60.7.7.2.1 Resource Full Messages

The resource full message is generated when the branch/predicate history buffer is full or the sequential instruction counter overflows.

9. Only the Source that have the register written will generate a synchronization message.

10. A Watchpoint occurrence for one Engine generates synchronization message for that Engine only.

For messages caused by the branch/predicate history buffer filling, the current value of that buffer is transmitted as part of the resource full message. This information can be concatenated by the development tool with the branch/predicate history information from subsequent messages to obtain the complete branch history for a message. The HIST value is reset by this message, and the I-CNT value is reset as a result of a bit being added to the history buffer.

For messages caused by sequential instruction counter overflow, the HIST packet is also transmitted. This occurrence indicates that exactly 256 sequential instructions have been executed without a history bit being recorded or a message (other than CHAN_WRITE) being generated. The HIST and I-CNT values are reset by this message.

The resource full message has the format shown in the following figure.

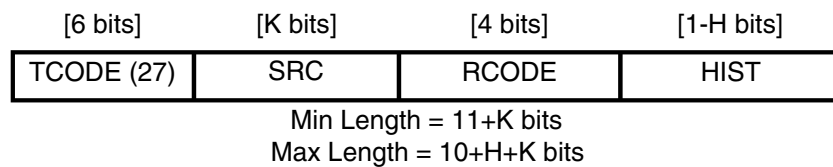


Figure 60-38. Resource Full Message Format

60.7.7.2.2 Indirect Branch with History Messages

The indirect branch with history message is generated on a taken indirect branch with no synchronization event pending. The program trace indirect branch with history message has the format shown in the following figure.

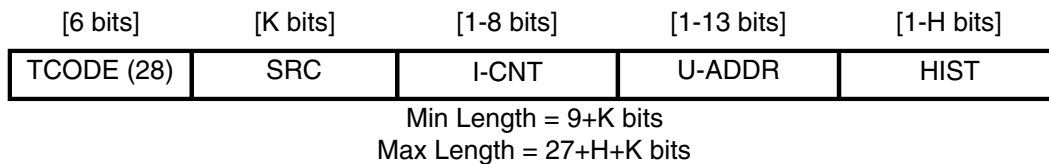


Figure 60-39. Indirect Branch with History Message Format

60.7.7.2.3 Indirect Branch with History Synchronization Messages

The indirect branch with history synchronization message is generated on a taken indirect branch with a synchronization event pending.

The program trace indirect branch with history synchronization message has the format shown in the following figure.

[6 bits]	[K bits]	[1-8 bits]	[1-13 bits]	[1-H bits]
TCODE (29)	SRC	I-CNT	F-ADDR	HIST

Min Length = 9+K bits
Max Length = 27+H+K bits

Figure 60-40. Indirect Branch with History Synchronization Message Format

60.7.7.2.4 Program Trace Correlation Message

The program trace correlation message is generated at the end of an ETPU channel service if there is no new service, if there is a new service which channel is not enabled for tracing, if program trace is disabled in the middle of a thread being traced, upon entry into debug mode, or upon entry into low power mode.

The message contains the history and sequential instruction count information generated during the previous traced service since the last message generation.

The correlation message also contains an event code that indicates the event responsible for the generation of the message. Event code encodings are shown in the following table.

Table 60-32. Event Code Encodings

EV-CODE	Description
0b0000	Entry into Debug Mode
0b0001	Entry into Low Power Mode
0b0010	Program Trace disabled in the middle of a thread
0b1110	End Channel Service (no new service)
all others	Reserved

The program trace correlation message has the format shown in the following figure.

[6 bits]	[K bits]	[4 bits]	[1-8 bits]	[1-H bits]
TCODE (33)	SRC	EV-CODE	I-CNT	HIST

Min Length = 12+K bits
Max Length = 18+H+K bits

Figure 60-41. Program Trace Correlation Message Format

60.7.7.2.5 Channel Start Service Message

The channel start service message is generated at the start of an ETPU channel service. The message contains the history and sequential instruction count information generated during the previous traced service since the last message generation. The message also contains the address of the first instruction of the current service.

The program trace channel start service message has the format shown in the following figure.

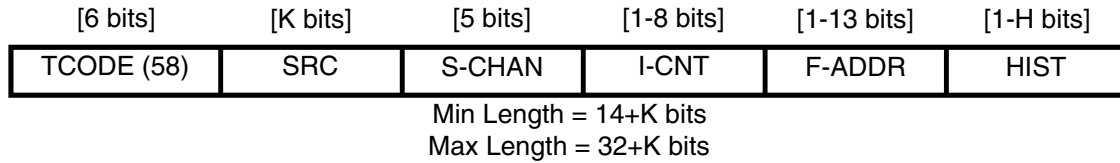


Figure 60-42. Channel Start Service Synchronization Message Format

60.7.7.2.6 Channel Trace Enable Message

If the program trace is enabled in the middle of a thread or after the Queue is enabled for storing snapshots and it was previously disabled due to queue overrun, the Channel Trace Enable Message is sent indicating to the development tool the number of the channel being serviced and the address of the current instruction.¹¹ If the CHAN register value differs from the serviced channel, the CHAN field is sent at the end of the message indicating the current CHAN register value.¹²

The program trace channel trace enable message has the format shown in the following figure.

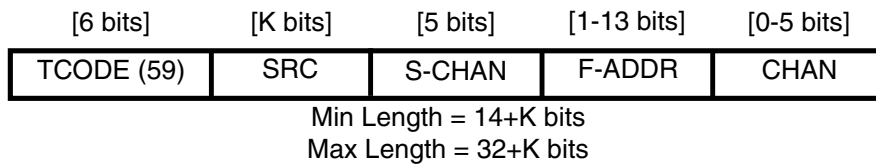


Figure 60-43. Channel Trace Enable Synchronization Message Format

60.7.7.2.7 Channel Register Write Messages

The channel register write message is generated when the CHAN register is written to by the microcode. This message is not necessary to reconstruct program flow and generation is enabled/disabled by the CHW bit in the ENGINE_x development control register. Program trace must be enabled for this message to be generated.

The channel register write message has the format shown in the following figure

-
11. The message does not contains the history and sequential instruction count information since the trace of this channel was disabled until there.
 12. Usually the CHAN register equals the serviced channel.

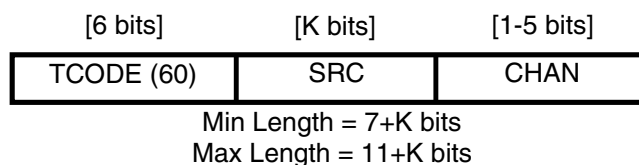


Figure 60-44. Channel Register Write Message Format

60.7.7.3 Branch Trace Messaging Operation

Figure 60-45 and Figure 60-46 show the basic flow used to generate branch trace messages for the ETPUs. The following acronyms are used in the figures:

- CHAN - CHAN register in ETPU execution unit
- S-CHAN - Channel being serviced
- HIST - Nexus shift register whose value is transmitted in the HIST packet of many BTMs
- I-CNT - Nexus sequential instruction counter whose value is transmitted in the HIST packet of many BTMs
- CWE - Channel Write Event
- IHE - Indirect Branch with History Event
- IHS - Indirect Branch with History Synchronization Event
- CSE - Channel Start Service Event
- CTE - Channel Trace Enable Event
- RFE - Resource Full Event
- PCE - Program Trace Correlation Event

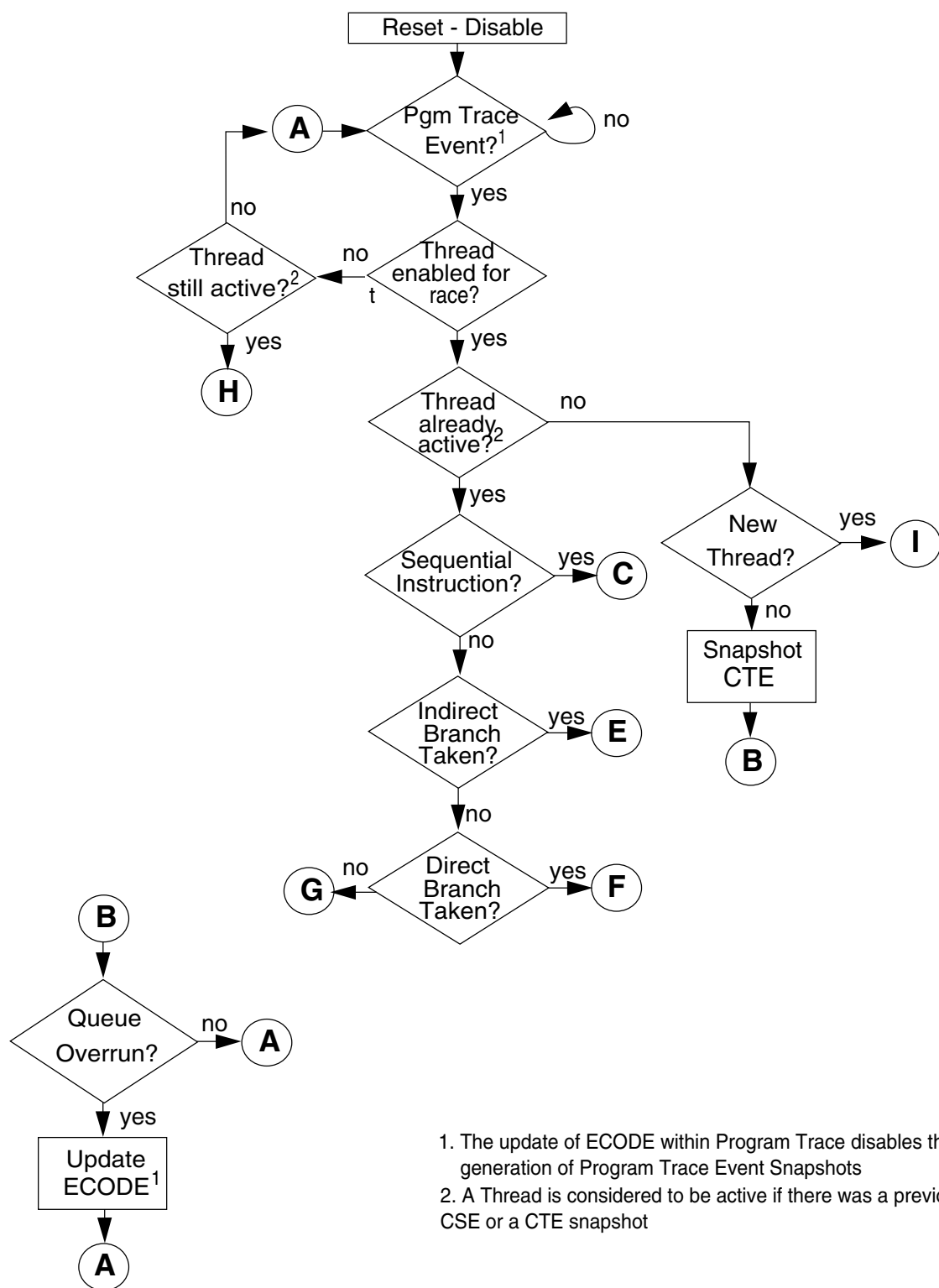


Figure 60-45. Branch Trace Message Generation (Part 1)

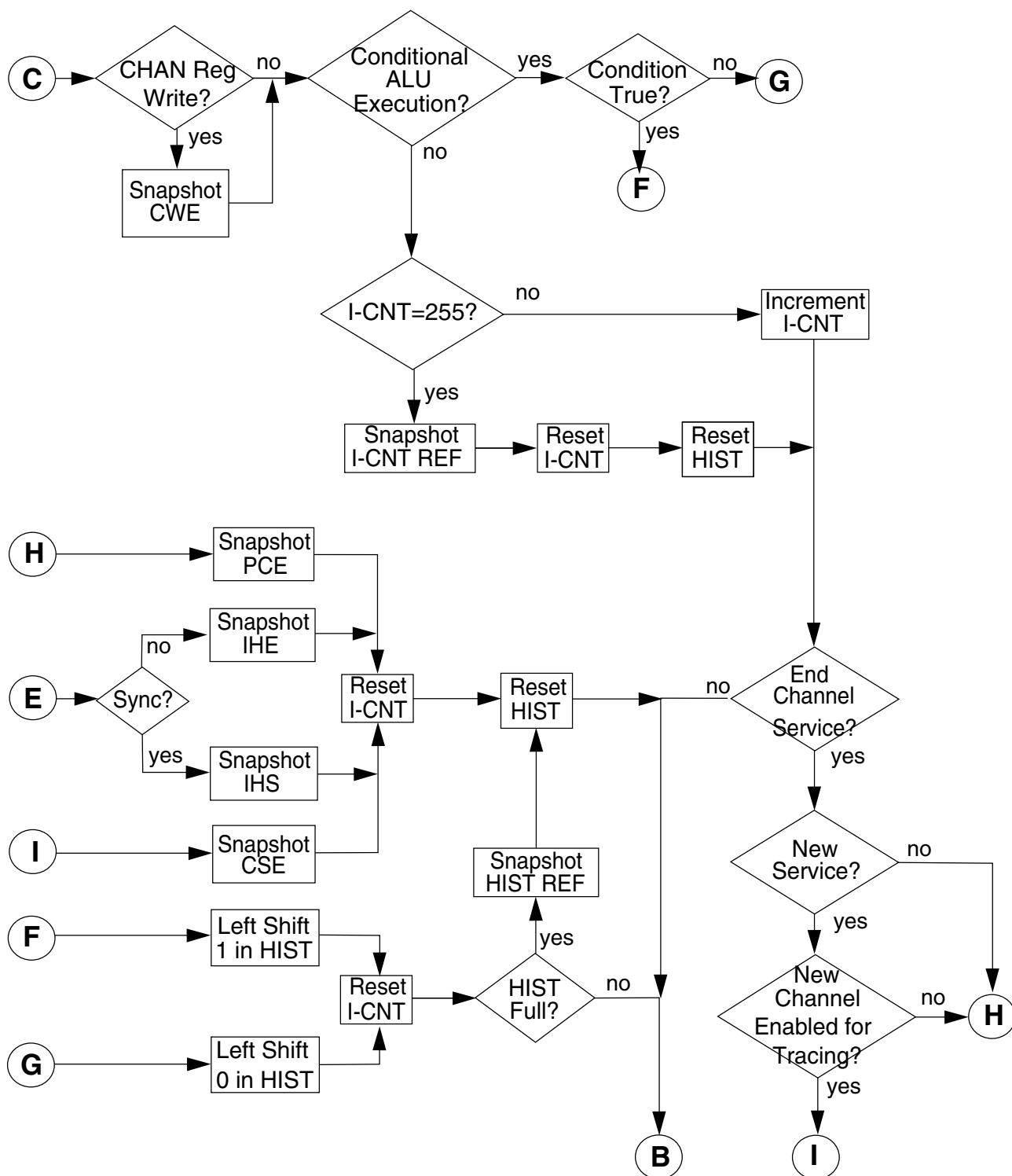


Figure 60-46. Branch Trace Message Generation (Part 2)

60.7.7.3.1 Relative Addressing

The relative address feature is compliant with the IEEE-ISTO 5001-2002 standard recommendations and is designed to reduce the number of bits transmitted for addresses of program trace messages.

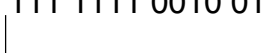
The address transmitted for a determined source is relative to the address of the previous branch trace message sent for that source. It is generated by XORing the new address with the previous address, and then using only the results up to the most significant '1' in the result. To recreate this address, an XOR of the (most-significant 0-padded) message address with the previously decoded address gives the current address. The following figure shows how a relative address is generated and how it can be used to recreate the original address.

Engine 1: Current Address (A1) = 0x65DC,
 Engine 2: Current Address (B1) = 0x7238

Next Message Relative to Engine 1: Address (A2) = 0x1AF8

Address Generation:

A1 = 110 0101 1101 11 00
 A2 = 001 1010 1111 10 00
 A1 ^ A2 = 111 1111 0010 01 00



Address Message (M1) = 111 1111 0010 01

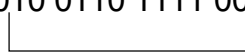
Address Recreation:

A1 ^ M1 = A2
 A1 = 110 0101 1101 11 00
 ME1 = 111 1111 0010 01
 A1 ^ ME1 = 001 1010 1111 10 00 = A2

Next Message Relative to Engine 2: Address (B2) = 0x54C8

Address Generation for Engine 2

B1 = 111 0010 0011 10 00
 B2 = 101 0100 1100 10 00
 B1 ^ B2 = 010 0110 1111 00 00



Address Message (ME2) = 10 0110 1111 00

Address Recreation:

B1 ^ M1 = B2
 B1 = 111 0010 0011 10 00
 ME2 = 010 0110 1111 00
 B1 ^ ME2 = 101 0100 1100 10 00 = B2

Figure 60-47. Relative Address Generation and Recreation

60.7.7.3.2 Enabling Program Trace

Program trace for the ETPU is enabled on a channel-by-channel basis. This selective enabling allows the tool more control over how much information gets transmitted.

For program trace information to be generated for a channel, the appropriate bit in the program trace channel enable register must be set and program trace must be enabled via the TM field in the DC register or via a watchpoint trigger.

Any combination of channels can be enabled for trace at the same time by configuring the program trace channel enable register. See [ENGINEx Program Trace Channel Enable Register \(NDEDI_ENGINEx_PTCE\)](#) for more details.

Note that since the registers may be changed in the middle of a thread, program trace may be enabled or disabled in the middle of a thread.

If program trace for a determined channel is enabled in the middle of a service routine, a Channel Trace Enable Message is sent to the development tool indicating the number of the channel being serviced and the address of the current instruction. Otherwise, normal program trace begins only on the following service for that channel.

If program trace for a determined channel is disabled in the middle of a service routine, a Program Correlation Message is sent to the development tool indicating that the trace is disabled and informing the HIST and I-CNT values.

60.7.7.3.3 Branch/Predicate Instruction History

The HIST packet provides a history of direct branch and predicated instruction execution used for reconstructing the program flow. This packet is implemented as a left-shifting shift register. The register is always pre-loaded with a value of 1. This bit acts as a stop bit so the tool can determine which bit is the end of the history information. The pre-loaded 1 bit itself is not part of the history information but is transmitted with the packet.

A value of 1 is shifted into the HIST packet on a taken direct branch (conditional or unconditional) and on a predicate instruction whose predicate condition was evaluated as true. A value of 0 is shifted into the HIST packet on a not taken direct branch and on a predicate instruction whose predicate condition was evaluated as false.

A direct branch is a branch whose target address is hard-coded into the microcode instruction at compile time. An indirect branch is a branch whose target address depends on run time conditions. The indirect branches in the ETPU are return from subroutine, dispatch jump, and dispatch call.

For the ETPU, predicated instructions are those that execute ALU operations conditionally based on the value of the C, Z, or N flags. This conditional execution is controlled by the value of the AS/CE packet in the ETPU microcode instruction formats A3, B4, B5, and B6.

60.7.7.3.4 Sequential Instruction Count

The I-CNT packet, present in many of the program trace messages, represents the number of ETPU microcode instructions executed since the start of channel service, last taken direct branch, last not-taken direct branch, last taken indirect branch, or last predicated instruction.¹³ In other words, I-CNT is reset whenever it is transmitted within a message or a bit is recorded to HIST.

When a sequential instruction is executed and the current I-CNT value is 255, a resource full message is transmitted to tell the tool that the I-CNT has reached a value of 256. The current HIST packet is transmitted as part of this message and reset inside the NDEDI. The HIST and I-CNT information in the resource full message are combined with information from subsequent messages to provide the development tool a full picture of the program flow.

If additional HIST bits are recorded between a resource full message for the I-CNT and the next program trace message, the I-CNT overflow can be ignored by the tool. In this case, the HIST packet transmitted by any resource full messages and the next program trace are concatenated to provide the full HIST information.

Multiple I-CNT resource full messages can be received between other program trace messages. In these cases, the HIST packets are concatenated, and the I-CNT values are added only when no HIST bits were recorded between the resource full messages.

60.7.7.3.5 Interleaved ENGINE1 and ENGINE2 messages

Both ENGINE1 and ENGINE2 can be enabled for program trace at the same time. Each Engine microcode instruction takes two system clocks to execute. The dual Engine system alternates instruction completion between ENGINE1 and ENGINE2, thus one module never completes an instruction in the same clock as the other.

The NDEDI block formats and queues ENGINE1 and ENGINE2 messages in the order they are generated. There is no way for the development tool to infer any other temporal information from the ETPU messages.

60.7.7.4 Timing Diagrams

Note that all of the following timing diagrams assume a 3-bit SRC size and either a 4-bit MDO port or a 16-bit MDO port. The state variable is not a signal, but instead is derived from `MSEO`. It is included for clarity. Refer to [Figure 60-23](#) for MSEO state diagram.

The following abbreviations are used for the state variable in the diagrams:

13. ETPU does not have a conditional indirect branch. Thus, there is no not-taken indirect branch.

- ID = Idle
- SM = Start Message
- NT = Normal Transfer
- EP = End Packet
- EM = End Message

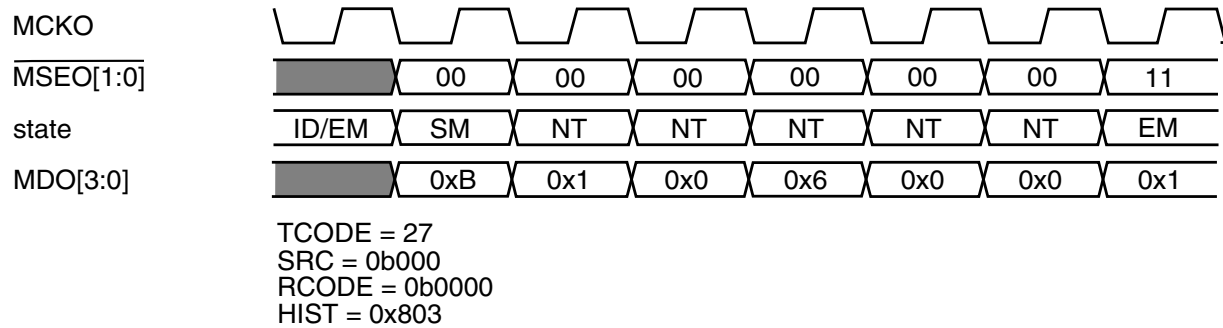


Figure 60-48. Resource Full Message, I-CNT Overflow

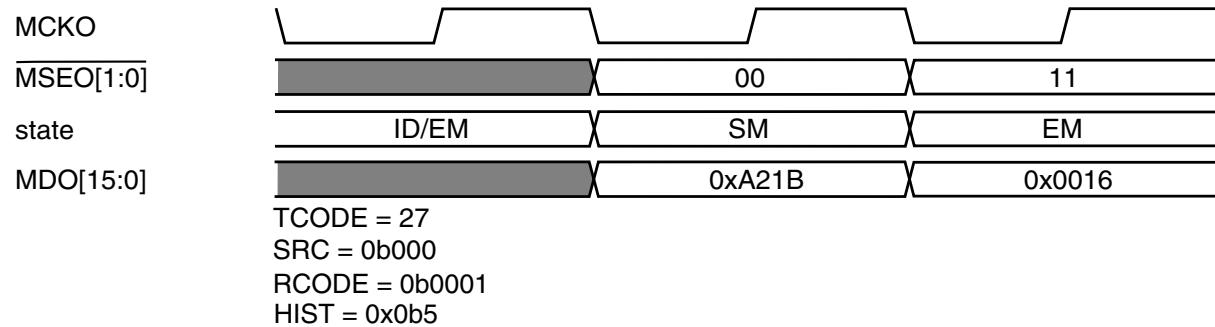


Figure 60-49. Resource Full Message, HIST Buffer Full

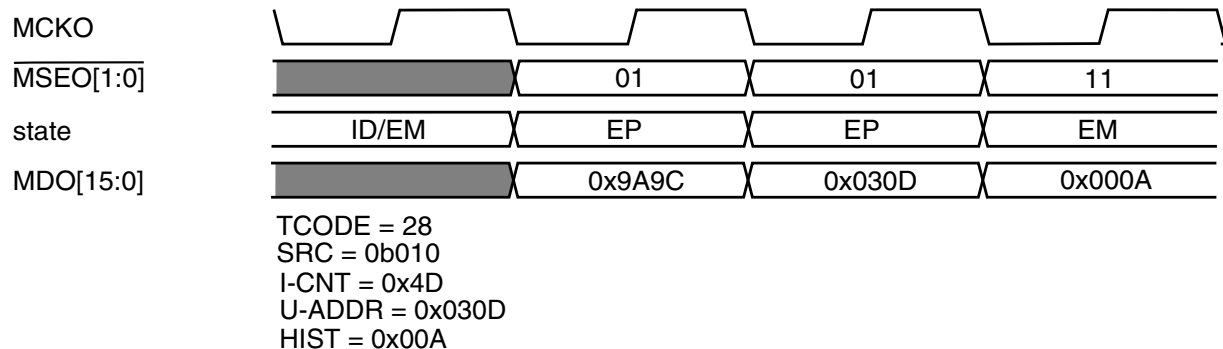
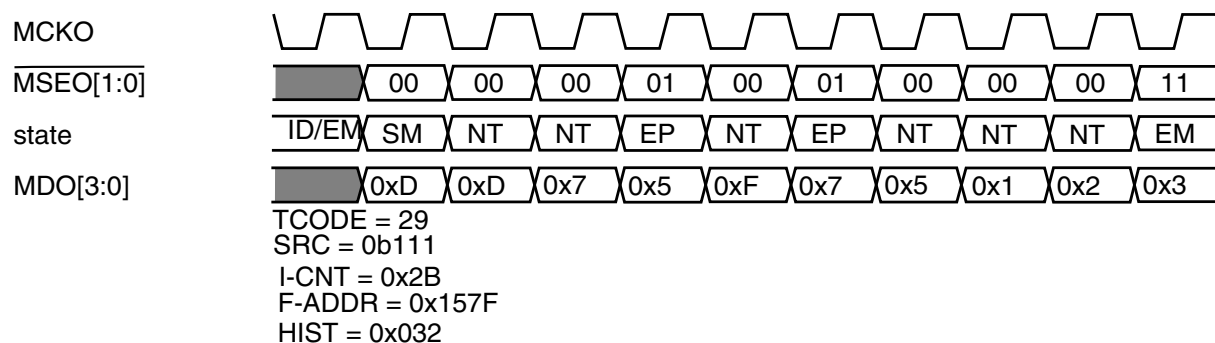
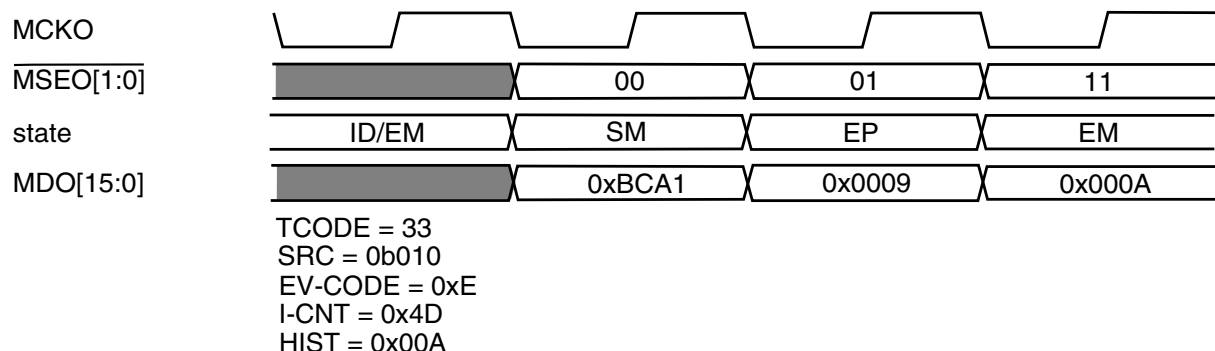
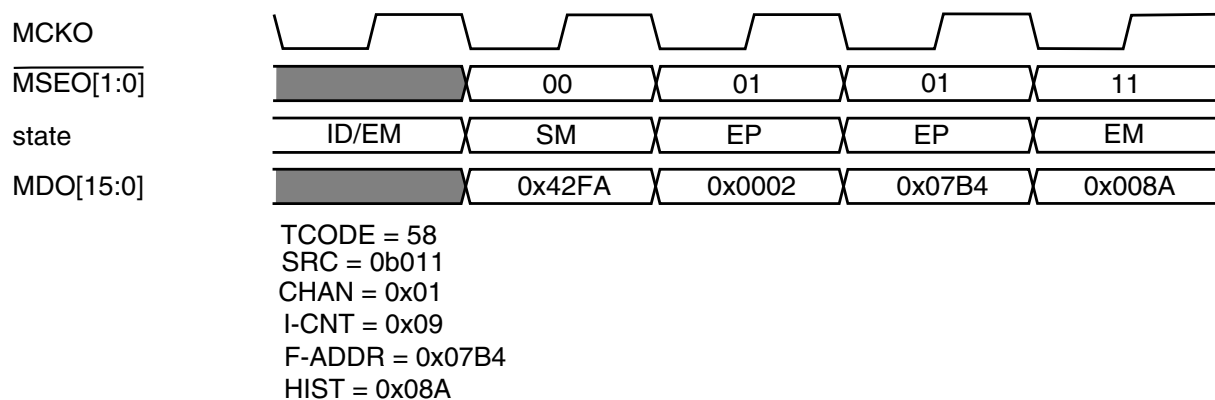
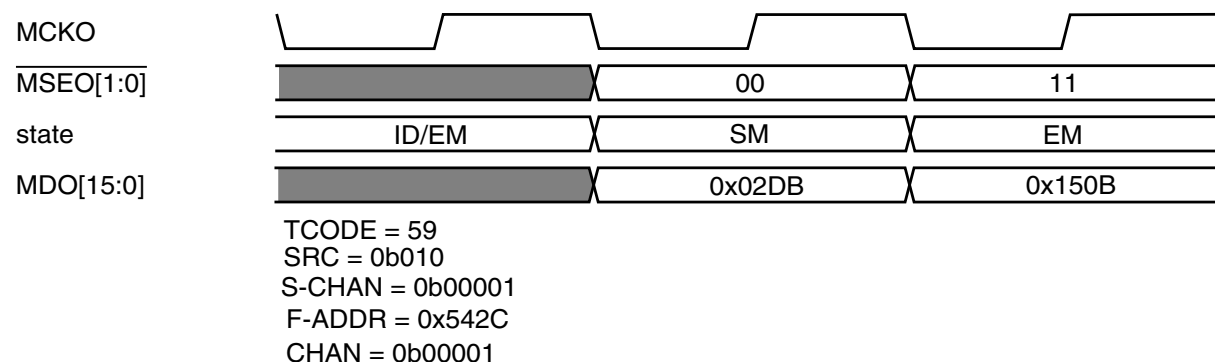


Figure 60-50. Indirect Branch with History Message

**Figure 60-51. Indirect Branch with History Synchronization Message****Figure 60-52. Program Trace Correlation Message****Figure 60-53. Channel Start Service Message****Figure 60-54. Channel Trace Enable Message with the same value for CHAN and S-CHAN**

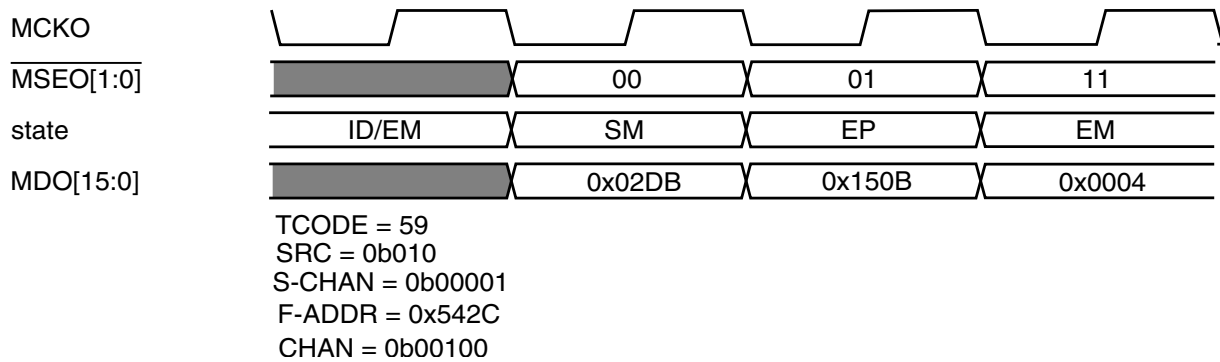


Figure 60-55. Channel Trace Enable Message with different values for CHAN and S-CHAN

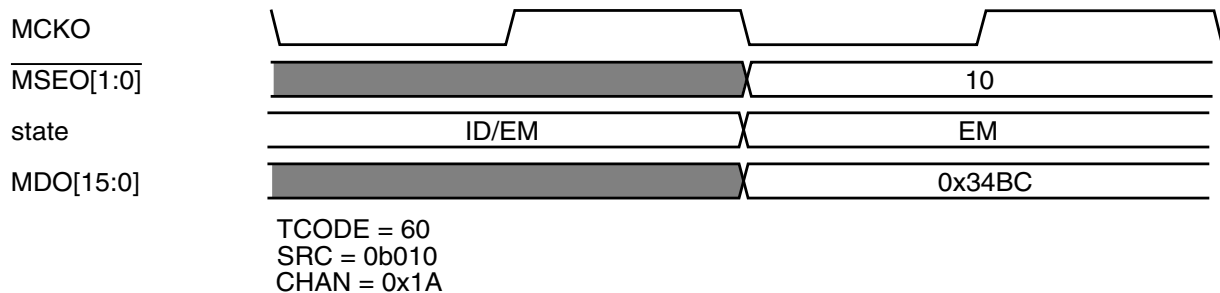


Figure 60-56. Channel Register Write Message

60.7.8 Data Trace

Dual Engines perform loads and stores to the Shared Parameter RAM (SPRAM). The NDEDI block gathers information about these accesses on a dedicated bus. The NDEDI block traces all SPRAM accesses that meet the selected address range and attributes. This includes accesses from either Engines as well as Coherent Dual-Parameter Controller (CDC) accesses. SPRAM accesses can have data sizes of 8, 24, and 32 bits. The NDEDI block supports all three data sizes.

Whenever performing 8-bits accesses, the ETPU accesses the 8 most significant bits of the word addressed, while the 24-bits accesses are accomplished by accessing the 24 least significant bits. Thus, since the ETPU system uses the Big Endian convention and since the ETPU data addresses are byte-relative the two least significant bits of the transmitted address follows the convention shown in the following table.

Table 60-33. Two least significant bits for Data Trace Addresses

Value	Access
0b00	ETPU 8-bit or 32-bit accesses
0b01	ETPU 24-bit accesses
0b1x	Are not meaningful for ETPU

60.7.8.1 Data Trace Message Formats

There are four types of Data Trace Messages (DTMs):

- Data Write Message
- Data Read Message
- Data Write with Synchronization Message
- Data Read with Synchronization Message

All of the data trace messages contain a SIZE packet. The following table shows the decoding for this packet.

Table 60-34. Data Trace SIZE Packet Decodings

SIZE Packet	Access Size
0b00	Not Meaningful
0b01	24 bits
0b10	8 bits
0b11	32 bits

60.7.8.1.1 Data Write Message

The data write message contains the data write value and the address of the target location relative to the address of the previous data trace message from the same source.

The data write message has the format shown in the following figure.

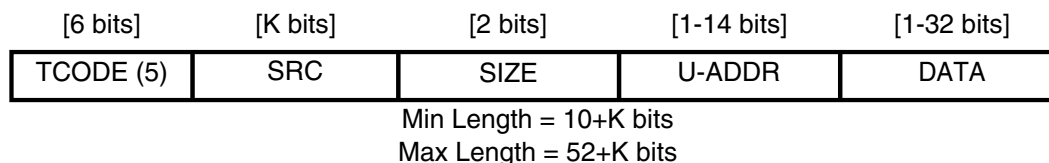
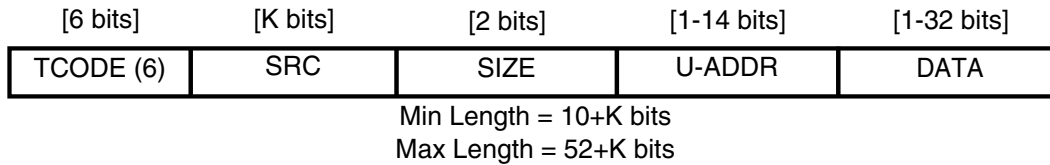


Figure 60-57. Data Write Message Format

60.7.8.1.2 Data Read Message

The data read message contains the data read value and the address of the target location relative to the address of the previous data trace message from the same source.

The data read message has the format shown in the following figure.

**Figure 60-58. Data Read Message Format**

60.7.8.1.3 Data Trace Synchronization Messages

The occurrence of any of the following conditions requires the following data trace message to be synchronizing:

- Exit of system reset
- Exit of low power mode
- Exit of debug mode
- Write to the development control register or data trace control register¹⁴
- Write to any of the ETPU data trace address range registers
- Exit of NDEDI reset
- 255 data trace messages of the same source are queued without synchronization
- Assertion of the Event In ($\overline{\text{EVTI}}$) pin, depending on how the development control register is configured
- A watchpoint occurrence¹⁵
- A queue overrun

Data trace synchronization messages provide the full addresses (without leading zeros) and ensure that the development tools fully synchronize with data trace regularly. Each synchronization message provides a reference address for subsequent DTMs, in which only the unique portion of the data trace address is transmitted.

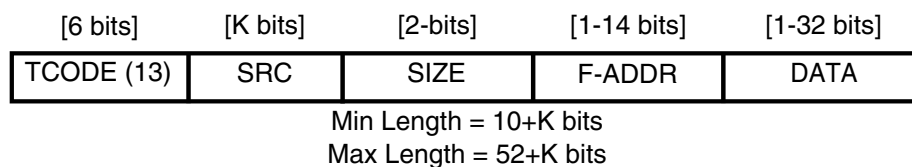
There are two types of data trace synchronization messages: data write synchronization and data read synchronization.

60.7.8.1.3.1 Data Write Synchronization Message

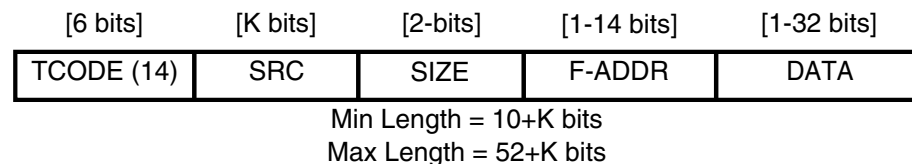
The data write synchronization message has the format shown in the following figure.

14. Only the Source that have the register written will generate a synchronization message.

15. A Watchpoint occurrence for one Engine generates synchronization message for that Engine only.

**Figure 60-59. Data Write Synchronization Message Format****60.7.8.1.3.2 Data Read Synchronization Message**

The data read synchronization message has the format shown in the following figure.

**Figure 60-60. Data Read Synchronization Message Format****60.7.8.2 Data Trace Operation**

Data tracing is performed by snooping a dedicated ETPU/Nexus interface for SPRAM read and write cycles. Data trace functions are enabled by setting the appropriate fields in the following registers:

- ENGINE1 Development Control (NDEDI_ENGINE1_DC) Register
- ENGINE2 Development Control (NDEDI_ENGINE2_DC) Register
- ENGINE1 Data Trace Control (NDEDI_ENGINE1_DTC) Register
- ENGINE2 Data Trace Control (NDEDI_ENGINE2_DTC) Register
- CDC Data Trace Control (NDEDI_CDC_DTC) Register
- ETPU Data Trace Address Range 0 (NDEDI_ETPU_DTAR0) Register
- ETPU Data Trace Address Range 1 (NDEDI_ETPU_DTAR1) Register
- ETPU Data Trace Address Range 2 (NDEDI_ETPU_DTAR2) Register
- ETPU Data Trace Address Range 3 (NDEDI_ETPU_DTAR3) Register

For details on register configuration, refer to [Register descriptions](#).

The dual Engines and CDC data tracing share the same Data Trace Address Range Registers. This enables usage of more address ranges for one client when the others are not being traced, or monitoring a certain range regardless of which client accesses the range.

The ETPU provides information to the NDEDI block when the access is actually being performed. The NDEDI does not have to track a bus protocol or check for error conditions on the accesses, since SPRAM does not generate errors. Any access signaled by the ETPU/NDEDI interface completes without error. Data trace flow is depicted in the following figure.

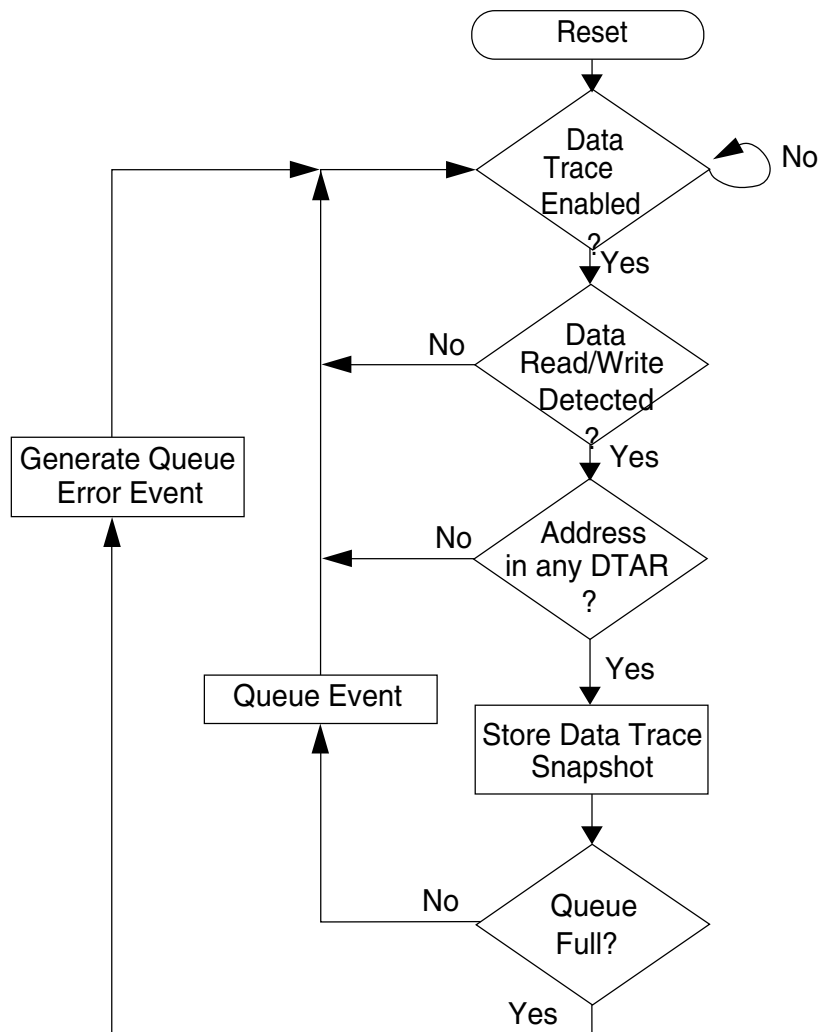


Figure 60-61. ETPU/CDC Data Trace Flow Diagram

60.7.8.2.1 Data Trace Windowing

Data trace windowing is provided so the development tool can decrease the auxiliary port usage by limiting the accesses that are traced.

Data trace windowing is achieved via the address range defined by the DTSA and the DTEA fields of the DTAR registers. These registers are shared by ENGINE1, ENGINE2, and the CDC. SPRAM accesses will be traced if their address fall in any of these ranges and the specific range is enabled in the data trace control register of the source making the access. Data read and/or data write trace may be enabled via the Read/Write Trace Control fields in the data trace control registers.

Data trace ranges are 32-bit aligned. This alignment is done by making the two least significant bits of the DTSA and DTEA fields read only. Since the two least significant bits of DTSA and DTEA are read only and default to different values, DTSA can never equal DTEA. The following figure shows the relationship between the DTSA and DTEA fields.

Programmed Value	Ranged Selected
DTSA<DTEA	DTSA → ← DTEA
DTSA>DTEA	All addresses are out of Range ¹
DTSA==DTEA	Impossible ²

Figure 60-62. Data Trace Address Range Options

60.7.8.2.2 Relative Addressing

The relative address feature is compliant with the IEEE-ISTO 5001-2002 standard recommendations and is designed to reduce the number of bits transmitted for addresses of data read and data write messages.

The address transmitted for a determined source is relative to the address of the previous data trace message sent for that source. It is generated by XORing the new address with the previous address and then using only the results up to the most significant '1' in the result. To recreate this address, an XOR of the (most-significant 0-padded) message address with the previously decoded address gives the current address. The following figure shows how a relative address is generated and how it can be used to recreate the original address.

Previous Address (A1) = 0xC10, New Address (A2) = 0xF64

Address Generation:

A1 = 1100 0001 0000
 A2 = 1111 0110 0100
 A1 ^ A2 = 0011 0111 0100

Address Message (M1) = 11 0111 0100

Address Recreation:

A1 ^ M1 = A2
 A1 = 1100 0001 0000
 M1 = 0011 0111 0100
 A1 ^ M1 = 1111 0110 0100 = A2

Figure 60-63. Relative Address Generation and Recreation1

60.7.8.3 Timing Diagrams

Note that all of the following timing diagrams assume a 3-bit SRC size and either a 4-bit MDO port or a 16-bit MDO port. The state variable is not a signal, but instead is derived from $\overline{\text{MSEO}}$. It is included for clarity. Refer to [Figure 60-23](#) for $\overline{\text{MSEO}}$ state diagram.

The following abbreviations are used for the state variable in the diagrams:

- ID = Idle
- SM = Start Message
- NT = Normal Transfer
- EP = End Packet
- EM = End Message

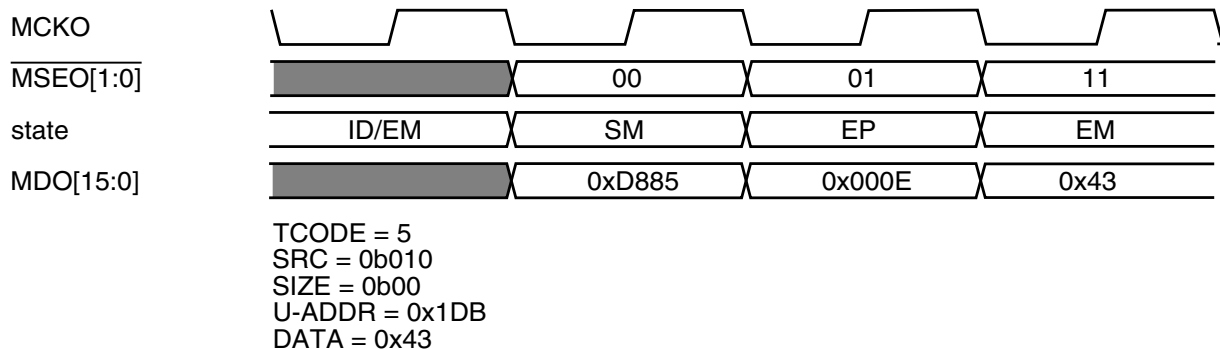


Figure 60-64. Data Write Message

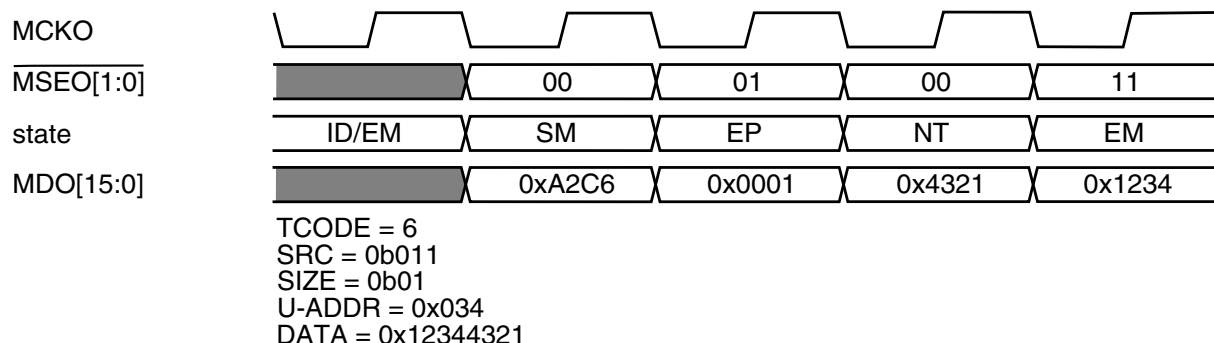


Figure 60-65. Data Read Message

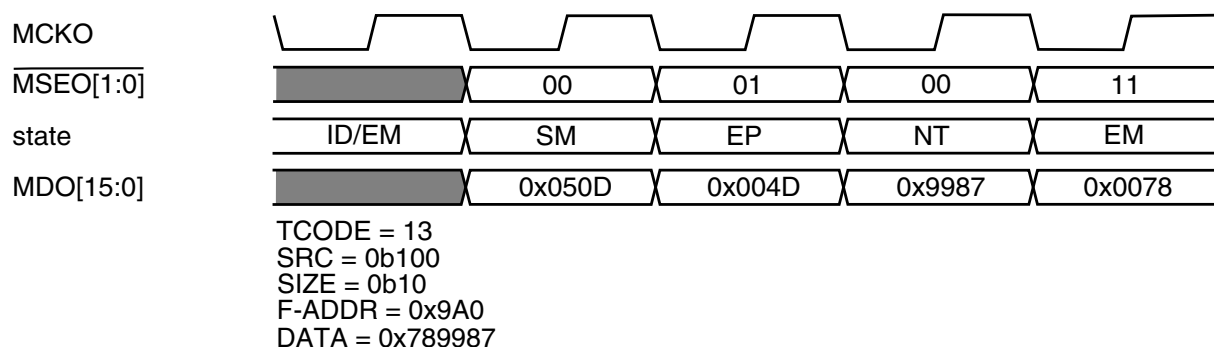


Figure 60-66. Data Write with Synchronization Message

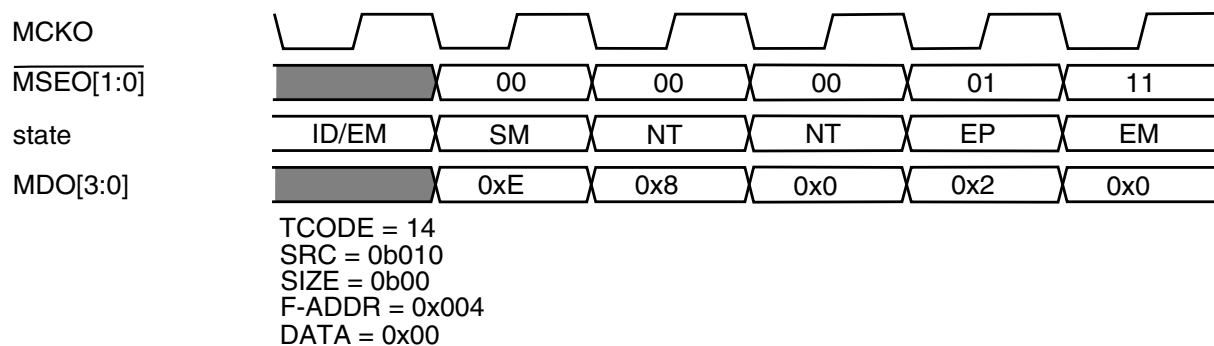


Figure 60-67. Data Read with Synchronization Message

60.7.9 Watchpoint Trace

The NDEDI module provides ETPU watchpoint messaging via the auxiliary port, as defined by the IEEE-ISTO 5001-2002 standard.

60.7.9.1 Messaging

The NDEDI block provides watchpoint messaging using IEEE-ISTO 5001-2002 standard-defined public messages. When a watchpoint occurs, a watchpoint event is sent to the Event Queue. If the watchpoint condition occurs while the Event Queue is not enabled for storing snapshots, a watchpoint overrun message is generated.

The watchpoint message has the format shown in the following figure.

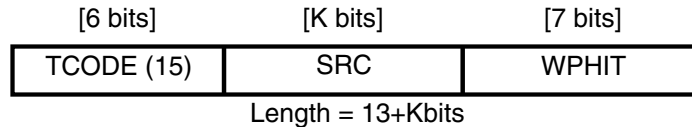


Figure 60-68. Watchpoint Hit Message Format

The values for the WPHIT packet are described in [Table 60-35](#) . Notice that WPHIT is never 0 because the message is only generated when a watchpoint has occurred.

Table 60-35. WPHIT Values

Value	Meaning
0b1xxxxxx	ETPU Watchpoint 1 (based on BWC1 register)
0bx1xxxxx	ETPU Watchpoint 2 (based on BWC2 register)
0bxx1xxxx	Channel Register Write Watchpoint
0bxxx1xxx	Host Service Request Watchpoint
0bxxxx1xx	Link Register Watchpoint
0bxxxxx1x	MRL Watchpoint
0bxxxxxx1	TDL Watchpoint

60.7.9.2 Error Messages

A watchpoint overrun error event is queued if an watchpoint event occurs and the Event Queue is not enabled for storing snapshots. This event is queued as soon as the Event Queue gets empty.

The error message has the format shown in the following figure.

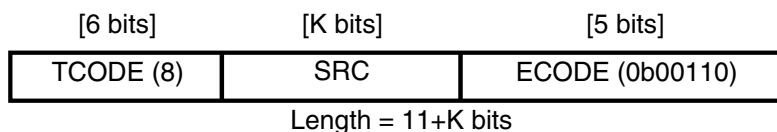


Figure 60-69. Watchpoint Overrun Error Message Format

60.7.9.3 Synchronization

Upon the occurrence of a watchpoint, the next program and data trace messages are synchronization messages. A watchpoint in one of the Engines causes data and program synchronization for that Engine only.

60.7.9.4 Timing Diagrams

Note that all of the following timing diagrams assume a 3-bit SRC size and a 16-bit MDO port. The state variable is not a signal, but instead is derived from $\overline{\text{MSEO}}$. It is included for clarity. Refer to [Figure 60-23](#) for $\overline{\text{MSEO}}$ state diagram.

The following abbreviations are used for the state variable in the diagrams:

- ID = Idle
- SM = Start Message
- NT = Normal Transfer
- EP = End Packet
- EM = End Message

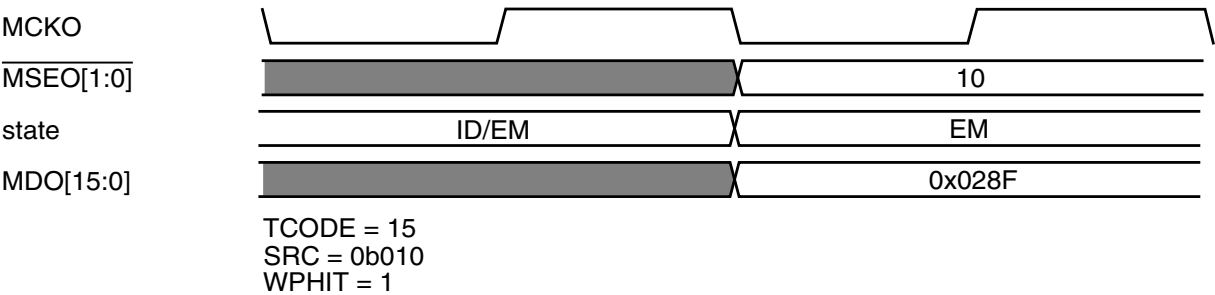


Figure 60-70. Watchpoint Message

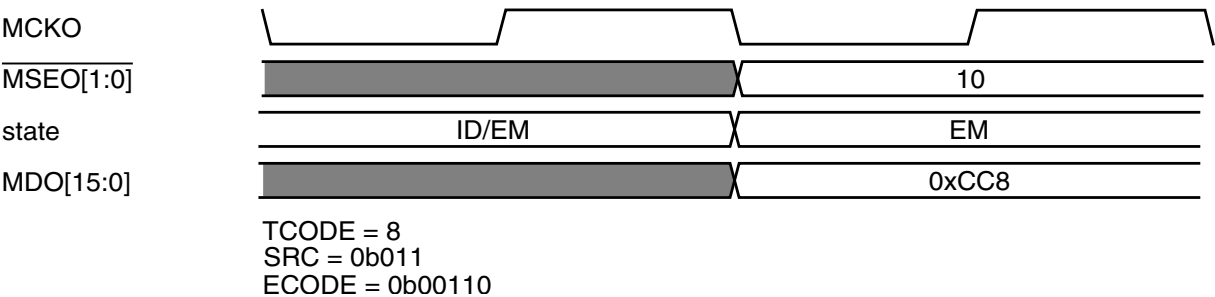


Figure 60-71. Watchpoint Overrun Error Message

60.7.10 ETPU Message Queue

The NDEDI implements an external Q_SIZE location Queue that is shared by data trace, ownership trace, program trace, watchpoint, error and debug status events.

One single microcycle may generate a combination of data trace, ownership trace, program trace, watchpoint, error and debug status events. The data trace events related to one engine have different timing though, being generated at the system clock subsequent to the other events. Thus, since the ENGINE1 and ENGINE2 alternate instructions, a snapshot may contain data trace information regarding to one engine and other events information related to the other engine.

With the occurrence of one or more events at one specific cycle, a snapshot of the signals, that can give information about all events, is queued in only one location of the Queue. The information of the source which have generated the events is also queued at the same position, so the Message Formatter can reconstruct the messages of the occurred events. [Figure 60-72](#) shows the Event Queue Block Diagram.

The Queue operation is transparent to the user.

The Event Queue is read whenever there is at least one position of the FIFO occupied and there are no concurrent writes to the Queue. Thus, the appropriate message is formatted and a request to send the message is sent to the Nexus Port Controller (NPC). The request will be assigned a level based on the number of used positions in the Queue as shown in the following table.

Table 60-36. MDO Request Level

Level	Used positions (q) as a fraction of the Queue size
0	$q = 0$
1	$1 \leq q < 0.5$
2	$0.5 \leq q < 0.75$
3	$0.75 \leq q^1$

1. The MDO request will also be 3 whenever the Queue is disabled for storing new snapshots due to an overrun.

According to the request priority of the NDEDI and the other Nexus Blocks within the chip, the NPC will grant the Auxiliary port to the NDEDI, and as soon as the auxiliary port is granted, the NDEDI will send the message. If there are other pending messages in the same snapshot, a new request is sent and a new grant is expected from the NPC. The next snapshot is only read when there are no pending messages to be sent in the current snapshot and there are no concurrent write accesses to the Queue.

If more than one message event are to be sent, the Message Formatter sends first the Messages related to Debug Status, then the Messages related to Watchpoints, then the messages related to Program Trace and latest the Data Trace Messages.

If the Q_SIZE position of the Queue is full and an event occurs, the Event Queue is disabled for storing any information. The MDO request level will be 3 until the last snapshot is sent. When the Queue gets empty, an Error event is queued containing information of all lost events. At this point, snapshots may be queued again.¹⁶

NDEDI keep temporal ordering of messages relative to ENGINE1, ENGINE2 and CDC, but not relative to messages generated by other clients sharing the auxiliary port.

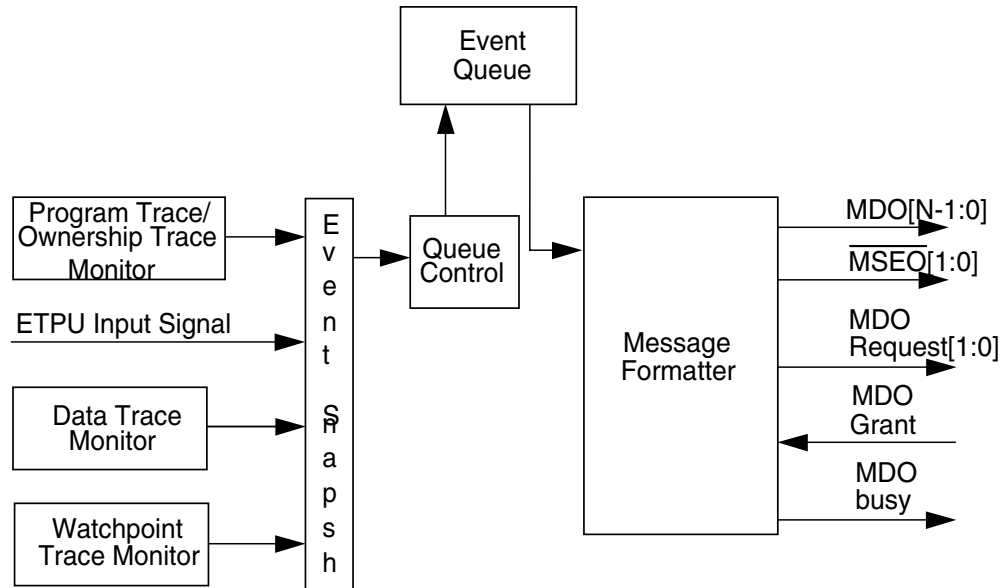


Figure 60-72. Event Queue

60.7.10.1 Queue Control

The queue is able to perform a single read or write operation per system clock. Thus, since there are no buffers to neither traces nor watchpoint events, the Message Formatter can only read the Event Queue if no events happen at the same time.

60.7.10.2 Error Messages

A trace or watchpoint overrun error occurs when an event cannot be queued. The error code (ECODE) within the error message indicates what type of trace overrun has occurred and that messages might have been lost for any or all of these events. [Table 60-25](#) shows the error code encodings for the various overrun error conditions.

16. If ENGINE_x is in stall condition, and an error happens, the ENGINE_x operation will only be resumed after the Queue is empty.

When information is lost, the Event Queue is disabled and no events are queued until the Event Queue gets empty. After the Event Queue gets empty an error message is queued indicating which information was lost for each source. At this point the Queue is enabled for storing snapshots again.

If multiple sources lose information at the same time multiple error messages are sent. In this case, the message related to the Engine 1 will be sent first, then the message related to the Engine 2 and finally the Error message related to the CDC.

The error message has the format shown in the following figure.

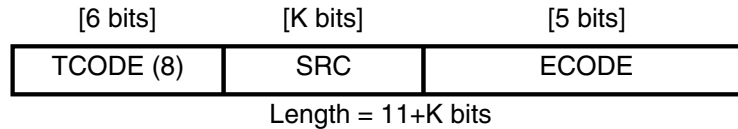


Figure 60-73. Error Message Format

60.7.10.3 Timing Diagrams

Note that the following timing diagram assumes a 3-bit SRC size and a 16-bit MDO port. The state variable is not a signal, but instead is derived from $\overline{\text{MSEO}}$. It is included for clarity. Refer to [Figure 60-23](#) for $\overline{\text{MSEO}}$ state diagram.

The following abbreviations are the values for the state variable in the diagram:

- ID = Idle
- SM = Start Message
- NT = Normal Transfer
- EP = End Packet
- EM = End Message

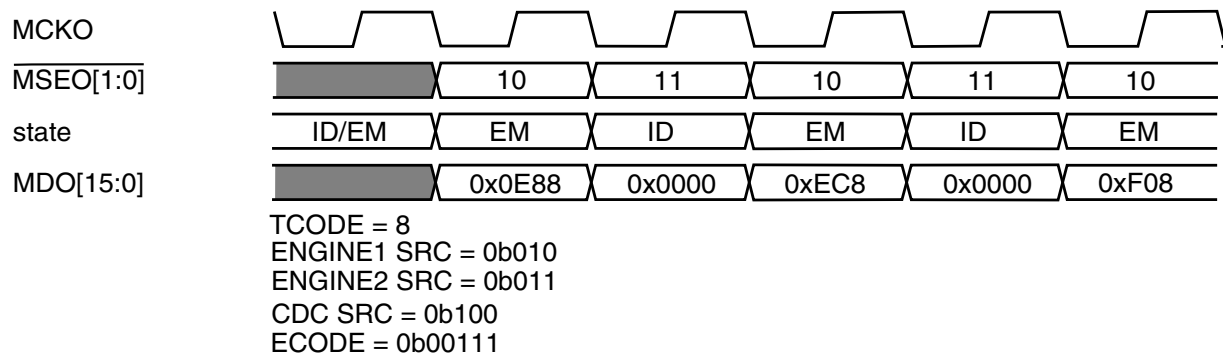


Figure 60-74. Error Messages (Program/Data/Ownership Trace Overrun)

Notice that three overrun messages are transmitted in the preceding figure. The first is an overrun indication for ENGINE1, the second is an overrun indication for ENGINE2, and the third is an overrun indication for the CDC.

60.8 Initialization/Application Information

The following section describes the Initialization/Application Information process.

60.8.1 Accessing NDEDI tool-mapped registers

To initialize the JTAG port for register accesses, the following sequence is required:

1. Enable the Nexus TAP controller
2. Retrieve Device ID if needed
3. Load the Nexus TAP controller with the NEXUS-ENABLE instruction

To write control data to NDEDI tool-mapped registers, the following sequence is required:

1. Write to the client select control register, if needed
2. Write the 7-bit register index and set the write bit to select register with a pass through the SELECT-DR-SCAN path in the JTAG state machine
3. Write the register value with a pass through the SELECT-DR-SCAN path. Notice that the prior value of this register is shifted out during the write

To read status and control data from NDEDI tool-mapped registers, the following sequence is required:

1. Write to the client select control register if needed
2. Write the 7-bit register index and clear write bit to select register with a pass through SELECT-DR-SCAN path
3. Read the register value with a pass through the SELECT-DR-SCAN path. Data shifted in is ignored

See [IEEE 1149.1 \(JTAG\) Input Port](#) for more detail.

60.8.2 Program Trace Reconstruction

This section details the program trace reconstruction mechanism supported by Nexus3 for the NDEDI. Program trace is implemented via Branch Trace Messaging (BTM) as per the Class 3 IEEE-ISTO 5001-2002 standard.

60.8.2.1 Branch Trace Messages

Branch Trace Messaging allows program trace by providing the following types of information:

- Messaging for start of program trace
 - Channel Start Message (CSM)
 - Channel Trace Enable Message (CTM)
- Branches with run-time target address (indirect branches)
 - Indirect Branch with History Message (IHM)
 - Indirect Branch with History and Synchronization Message (IHS)
- Discontinuity Information:
 - History including information of predicated instruction execution and information of all branches taken or not taken with compile-time target address. (direct branches)
 - Information of the amount of sequential instructions executed since the last branch. (ICNT)
 - Channel Start Message (CSM)
 - Indirect Branch with History and Synchronization Message (IHS)
 - Resource Full Message (RFM)
 - Program Correlation Message (PCM)
- Information of Program Trace End
 - Channel Start Message (CSM)
 - Program Correlation Message (PCM)
- Program trace overrun

- An Error Message will be issued if program trace information is lost due to FIFO overrun. The following error codes indicates program trace error: ECODE = 0b000001, 0b001111, 0b01000, 0b11001 and 0b11010

60.8.2.2 Assembly Code

The assembly code is necessary for gathering information about the direct branches location, target addresses for direct branches, predicated instructions location and "non-flush" indication.

The following eTPU instruction formats¹⁷ can generate direct branches:

E1, E2, E3 and E4

The following eTPU instruction formats can be predicated:

A3, B4, B5 and B6 but only if AS/CE differs 0, 1 and 7.

The following eTPU instruction formats can be indirect branches:

A1 with RTN = 1, D1, D2, D3 and D4

The Branch instructions have the capability of executing one instruction before changing the flow to the target address. The instruction being executed before the change of flow is hereafter called "non flushed" instruction.

The following eTPU instruction formats can lead to "non flushed" instructions:

D1, D2, D3, D4, E1, E2, E3 and E4 if FLS = 1

The other formats will lead to sequential instructions.

60.8.2.3 Branch Indication

The branch indication within the eTPU/NDEDI system happens at the instruction execution instead of the branch target execution. Thus it is important to notice that the "non flushed" instruction will be stored in the subsequent HIST/ICNT information. It is also important to notice that a CTM message is not issued if program trace is enabled in a "non flushed" instruction. In this case a CTM will be issued in the subsequent instruction.

17. Please refer to the ETPU Block Guide for further information about the instructions format.

60.8.2.4 Code Reconstruction

The CSM and CTM messages indicate the starting program trace address. Thus, these messages must be used as a starting point in order to synchronize the reconstruction program with the Assembly code.

From the starting point, the program flow can be completely reconstructed unless an Error message is issued. If an Error message is issued, watchpoints must be used to enable/disable program trace reducing the number of program trace messages.

60.8.2.5 Program Reconstruction Examples

Follows a code example where the reconstruction can be performed with the generated messages.

The program reconstruction will be performed by using the following two components:

- A copy of the program. Ideally this would be a listing file from the assembler;
- The program trace messages sent to the development tool;

60.8.2.5.1 Example 1

Follows the Example Code file:

```
234 FFFFFFFF format:F1.
235 010F64D1 format:A2 t2abd:"RAR", imm:578.
236 F7E846FF format:E4 j/c:"jump",bcc:"false",fls:"no_flush",bcf:"true",
    baf:567.
237 8AD4A218 format:B1 shf:"ror",cin:"no_carry",binv:"inv",t2abd:"P[31:24]",
    rw:"write",t4abs:"TRR",t4bbs:"BS=0,1st_2nd",src:">>1",p/d:"P",
    ccs:"flag_size",end:"yes",aid:24.
```

Follows the messages generated for a certain run example:

1. CSM (s=0x3) [S-CHAN=0x1e] [I-CNT=0x0000] [faddr=0x0234] [HIST=0x00000001];
2. PCM (s=0x3) [event=0x0] [I-CNT=0x01] [HIST=0x00000001];
3. CTM (s=0x3) [S-CHAN=0x001e] [faddr=0x0235] [CHAN=0x00000000];
4. PCM (s=0x3) [event=0x0] [I-CNT=0x00] [HIST=0x00000002];
5. CTM (s=0x3) [S-CHAN=0x001e] [faddr=0x0237] [CHAN=0x00000000];
6. CSM (s=0x3) [S-CHAN=0x11] [I-CNT=0x0001] [faddr=0x0443] [HIST=0x00000001];

Reconstruction:

The reconstruction starts with CSM pointing to the 0x0234 address. Note that ICNT = 0x0 and HIST = 0x1 indicating that there was no previous thread being executed.

PCM with event = 0x0 indicates that eTPU entered in debug mode. ICNT = 0x01 and HIST = 0x1 means that there were no direct branches and only one sequential instruction. Thus, instruction at address 0x0234 was executed and then eTPU entered in debug mode.

The third message is CTM which indicates that Program Trace was enabled again (eTPU exited debug mode) at address 0x0235. Then, another PCM with event = 0x0 was issued indicating entry in debug mode. From the code it is possible to inquire that instruction at address 0x0235 is a sequential instruction and the instruction at address 0x0236 is a conditional jump with "no flush".

HIST = 0b10 indicates that the first branch after address 0x0235 was reached but not taken. ICNT = 0x0 indicates that no sequential instruction after the last branch was executed, thus, it is possible to conclude that the code entered in debug mode just after executing the instruction at address 0x0236.

The fifth message is CTM which indicates a new Program Trace enable at address 0x0237.

The next message refers to CSM, and it indicates that a new thread is starting at address 0x0443. HIST=0b1 and ICNT=0b1 refers to the old thread and states that one sequential instruction (at address 0x0237) was executed before starting the new thread. This is the expected behavior since the address 0x0237 contains a "end=yes" command.

60.8.2.5.2 Example 2

Follows the Example Code file:

```
336 CB5CE283 3083 format:D1 mrle:"match_off",psc:"high",fls:"flush",rw:"write",
    pscs:"opac2",flc:"flags=P[25:24]",circ:"nop",
    r/d:"dispatch_call",p/d:"P",rsiz:"24",aid:131,
    end:"no".
...
41B 00AF6531 3962 format:A2 t2abd:"RAR", imm:1065.
41C F32083BF 3965 format:E4 j/c:"jump",bcc:"LSR",fls:"no_flush",bcf:"false",
    baf:1053.
41D 11092F98 3968 format:A1 t2d:"DIOB",rtn:"nop",IMM:9608768.
41E 004F7757 3971 format:B3 shf:"<<1",cin:"use_carry",binv:"inv",t4bbs:"SR",rw:"read",
    t4abs:"AS[23:0]=0",t2abd:"MACH",src:">>1",p/d:"DIOB",
    ccs:"no_flags",zro:"clear",end:"yes",
    stc:"no_inc/dec".
```

Follows the messages generated for a certain run example:

1. CSM (s=0x3) [S-CHAN=0x0f] [I-CNT=0x0001] [faddr=0x0336]
[HIST=0x0000000a];

2. IHM (s=0x3) [I-CNT=0x00] [uaddr=0x072d] [HIST=0x00000001];
3. CSM (s=0x3) [S-CHAN=0x14] [I-CNT=0x0003] [faddr=0x0345] [HIST=0x00000003];

Reconstruction:

The first CSM indicates that the thread will be started at the address 0x0336. The next message is an IHM with I-CNT = 0x0 and HIST = 0x1 indicating that no instructions were executed until the indirect branch. The message indicates u-addr = 0x072d, thus the target address of this instruction is 0x041b (0x0336 XOR 0x072d). Note that since the instruction at 0x0336 is a flush instruction, then the instruction at 0x0337 is not supposed to be executed.

The next message refers to CSM, and it indicates that a new thread is starting at address 0x0345. Since HIST = 0x3, either the next branch (after address 0x041b) was taken or the next predicated instruction (after address 0x041b) was executed. From address 0x041b the next branch instruction is located at 0x041c address, thus all instructions were executed until this address.

The instruction within the address 0x041c refers to a branch with no_flush instruction to address 0x041d.

Since I-CNT = 0x3, it is expected that 3 sequential instructions are executed after the branch instruction.

The instruction within 0x041d will be executed since it is a non-flushed instruction, then it will be executed again since 0x041d is the target for the branch instruction executed before, and finally the instruction within the address 0x041e is executed. Note that the instruction within 0x041e contains a "end=yes" command which will finish the current thread.

60.8.3 Microcode Development Support

The following sections describe the steps required to perform various class 1 features with the NDEDI block.

60.8.3.1 Read and Write SPRAM In Debug Mode

Reading and writing to the SPRAM is done through Nexus read/write access hardware external to the NDEDI block. Refer to the MCUs SoC Guide for details. SPRAM reads and writes are done the same in debug mode as they are in run mode.

60.8.3.2 Read and Write ETPU Internal Registers in Debug Mode

1. Select an area of the SPRAM to be used for dumping the register values
2. Read the current value of those SPRAM locations and store them
3. Write to the SPRAM as needed for writing internal ETPU registers
4. Execute external microcode that reads values from the SPRAM location and writes the values to registers or reads registers and writes the values to the SPRAM. See [Execute Forced Microcode Instruction in Debug Mode](#) for more details
5. Read the SPRAM as needed to retrieve register read values
6. Write the SPRAM back to the original values

60.8.3.3 Enter Debug Mode at the Negation of Reset

If the DBE field of the ETPU's Development Control register is asserted, NDEDI ETPU will enter debug mode as soon as the system reset is negated. Configure other bits as appropriate.

60.8.3.4 Enter Debug Mode During Normal Execution

To enter debug mode during normal execution, write the ETPU's Development Control register to set the DBE and DBR bits. Configure other bits as appropriate.

60.8.3.5 Stop Program Execution on a Breakpoint

There are four possibilities to stop the program execution on a Breakpoint:

1. Program execution halts at the end of the current microcycle when a synchronous breakpoint condition is reached. A synchronous breakpoint condition occurs due to the following breakpoints:
 - Breakpoint due to DBR assertion
 - Twin Engine breakpoint¹⁸

18. Depending on CBT value

- External breakpoint¹⁹
 - Breakpoint due to exit of system reset
 - breakpoint due to start of a channel service
 - Single Step breakpoint
 - breakpoint due to EVTI assertion
 - breakpoint due to MISC miscompare
2. Program execution stops at the beginning of the current microcycle when an asynchronous breakpoint condition is reached. An asynchronous condition occurs due to the following breakpoints:
 - breakpoint due to SPRAM access
 - breakpoint due to Illegal instruction
 - breakpoint due to CHAN register write
 3. Program execution stops at the end of the current thread if a twin breakpoint or an external source requests to enter in debug mode and the NDEDI is programmed to enter in debug mode only at the end of the current thread
 4. Whenever fetching an instruction, if the microprogram counter matches the BWA related to one of the breakpoint conditions programmed for breaking into an instruction fetch, the instruction is tagged indicating that it must not be executed. The microengine halts just before the execution of this instruction if it was supposed to be executed²⁰.

See [Set Breakpoint or Watchpoints](#) and [Register descriptions](#) for details on configuring breakpoints.

60.8.3.5.1 Multiple breakpoint conditions at the same address

The ETPU block must always execute the first instruction after exiting debug mode.

Assume that an engine is halted due to a synchronous breakpoint at a given address. As soon as the engine exits debug mode, the instruction at the given address will be executed anyway, even if other breakpoint conditions are matched, for instance tagged instructions or asynchronous breakpoints.

19. Depending on CBT value

20. Instructions flushed after a JUMP are not considered to be executed

Although the DS register doesn't signalize the other breakpoint conditions, the development tool can infer from the NDEDI registers and pre-loaded code if the engine is halted in one of these conditions.

60.8.3.6 Single Step Instructions and Re-Enter Debug Mode

To single step an instruction from shared code memory in debug mode, write to the Engine's Development Control register so the CBR bit is asserted, the DBR bit is cleared and the SS bit is asserted. Single stepping causes the Engine to exit debug mode for a microcycle where the TCRs, decrementors, and microprogram counter can update. Program Trace messages are generated within Single Step operation.

60.8.3.7 Set Breakpoint or Watchpoints

There are two main ETPU breakpoint/watchpoint sources that are independent of each other but have identical configuration registers. Each can be configured as either a breakpoint or watchpoint that detects reads and/or writes, compares data and/or address. Each can be configured to occur only on certain Channel services.

In addition, there is a breakpoint/watchpoint that can be configured to occur when the CHAN register is written to a certain value.

There are also four channel service breakpoint/watchpoints that can be configured to occur when a specific channel is serviced for one of the following reasons: host service request, link request, match detect, and transition detect.

Breakpoints cause the Engine to halt. Watchpoints cause a watchpoint hit message. All breakpoint/watchpoint sources can be configured to assert EVTO on an breakpoint/watchpoint occurrence.

Refer to [Register descriptions](#) for detailed descriptions of breakpoint/watchpoint registers.

60.8.3.8 Execute Forced Microcode Instruction in Debug Mode

To execute a forced instruction in debug mode, write the instruction into the Microinstruction debug register. This mechanism does not cause the Engine to exit debug mode so the microprogram counter is not updated (unless a branch instruction is executed) and the TCRs and decrementors are not updated if the CLKS bit is asserted in the Development Control Register. Executing instructions in this manner is useful for reading and writing internal Engine registers.

Chapter 61

Development Trigger Semaphore (DTS)

61.1 Introduction

The Development Trigger Semaphore (DTS) module enables software to signal an external tool by driving a persistent (affected only by reset or an external tool) signal on an external device pin. There are a variety of ways this module can be used, including as a component of an external real-time data acquisition system.

Note

When used as a component of a triggered data acquisition system, Nexus read/write access is different than the data acquisition protocol defined in the IEEE-ISTO 5001-2003 or IEEE-ISTO 5001-2012 Nexus standards, which use the Nexus Auxiliary port.

61.2 Overview

The Development Trigger Semaphore (DTS) module consists of registers and a small amount of combinational logic to generate an output signal—DTS Trigger Output (DTO). The registers are as follows:

- **DTS_SEMAPHORE** register—Any bit in this 32-bit register, when set to a value of logic '1', causes the DTS module output signal to be asserted if the corresponding **DTS_EN** bit is set in the **DTS_ENABLE** register. This enables an external tool to detect up to 32 signals from the application software. In an application, each bit is generally associated with a specific data set. Only the processor core and DMA module can set bits in this register. The bits can only be cleared by a tool access via Nexus Read/Write Access.

- **DTS_SEMAPHORE_B** register—Any bit in this 32-bit register, when set to a value of logic '1', causes the DTS module output signal to be asserted if the corresponding **DTS_EN_B** bit is set in the **DTS_ENABLE** register. This enables an external tool to detect an additional 32 signals from the application software. In an application, each bit is generally associated with a specific data set. Only the processor core and DMA module can set bits in this register. The bits can only be cleared by a tool access via Nexus Read/Write Access.
- **DTS_STARTUP** register—This register provides a mechanism for the external tool to notify software running on the CPU that the tool is connected and can provide information about either the type of tool or options that can be used by the software.
- **DTS_ENABLE** register—This register provides an enable/disable capability for the DTS feature.

The architecture is shown in the following figure.

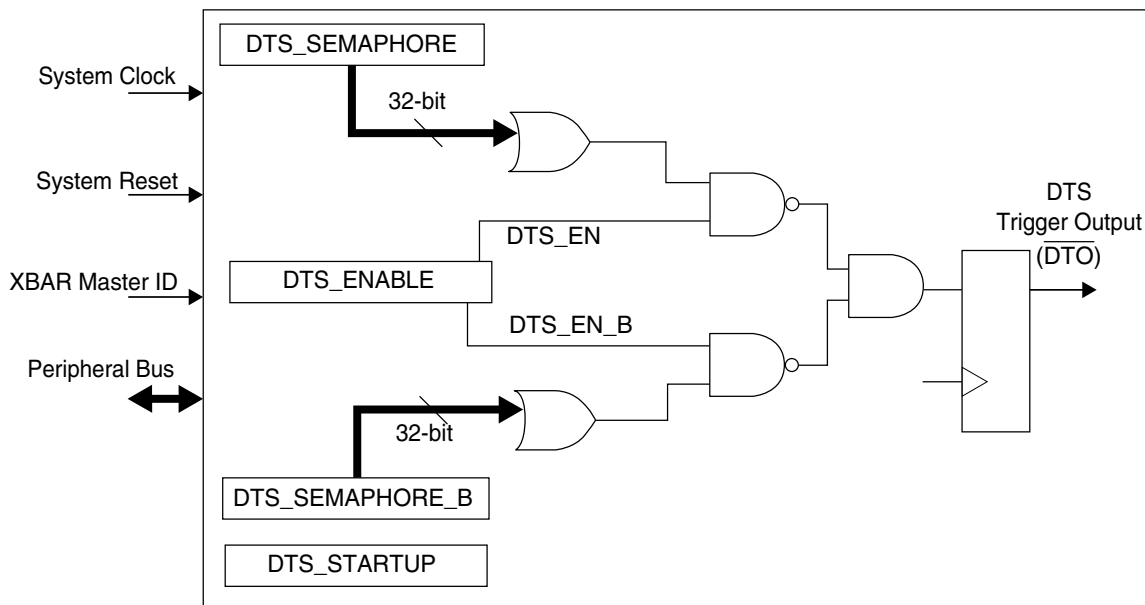


Figure 61-1. DTS block diagram

The DTS Trigger Output (DTO) signal is connected to one of the EVTO inputs of the Nexus Port Controller (NPC). The other EVTO inputs are connected to the other Nexus modules in the device. DTO is asserted when any bit in a semaphore register is set and the DTS function of that register is enabled.

Note

When the DTS module is enabled ($\text{DTS\_ENABLE}[\text{DTS\_EN}] = 0b1$ or $\text{DTS\_ENABLE}[\text{DTS\_EN\_B}] = 0b1$), all other Nexus EVTO functions associated should be disabled by the tool and

EVTO becomes the DTO. Unlike the EVTO function that only asserts for one clock, the DTO function remains asserted until the tool reads the semaphore register that is the source of the assertion, clearing the register's contents.

The following figure shows an example of the chain of events that begins with setting of a bit in the DTS_SEMAPHORE register and the clearing of the register caused by a Nexus read.

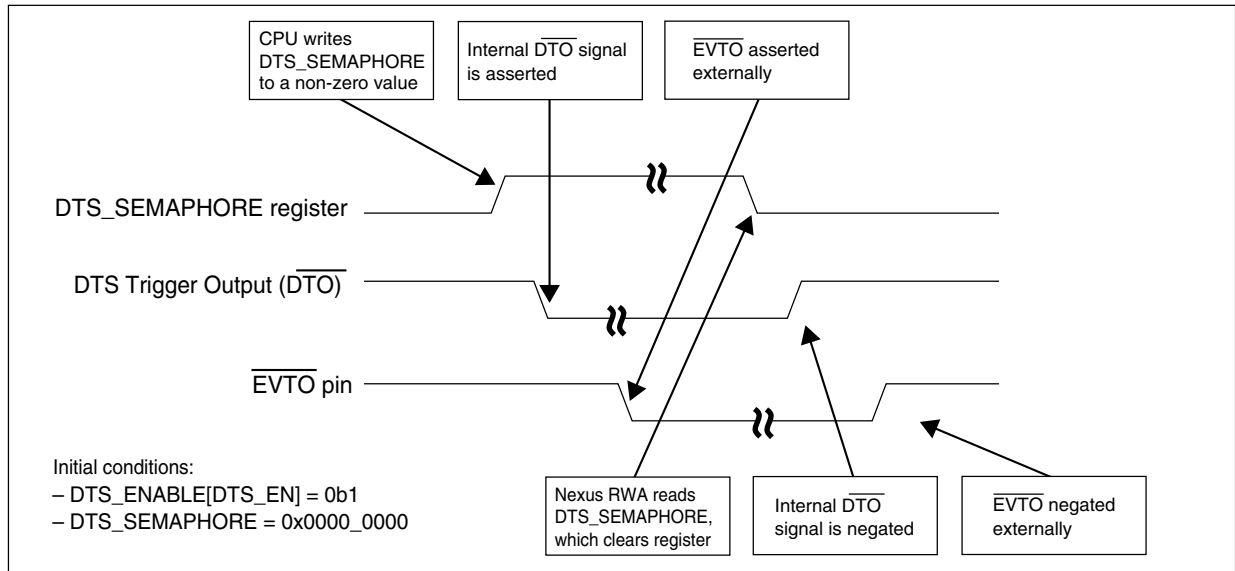


Figure 61-2. DTO event sequence example

61.3 DTS device connections

The following figure shows the DTS device connections. The DTS module connects to the Peripheral Bridge (PBRIDGE) for access to the registers. The PBRIDGE is connected to the device modules through a slave port of the Crossbar bus interface (XBAR).

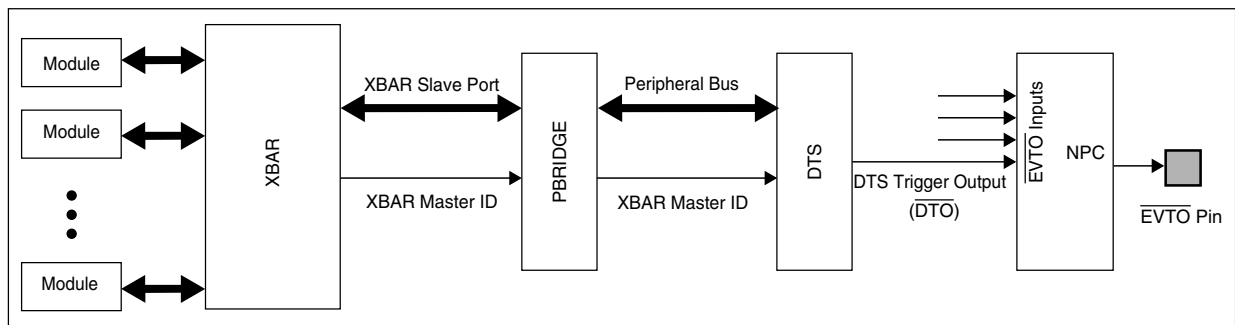


Figure 61-3. DTS device connections

The registers have limited access as described in [DTS register access](#). Access is based on the XBAR Master ID of the accessing module. Writing to the semaphore registers is limited to the cores and the eDMA module and is restricted to only setting bits. Only an access via a Nexus Read/Write Access from an external tool can clear bits in the semaphore registers (semaphore register bits are cleared automatically when read). Similarly, the DTS_ENABLE and DTS_STARTUP registers can only be written via a Nexus Read/Write Access.

Note

Nexus Read/Write Accesses use the core load/store bus to perform accesses, but Nexus accesses have a different Master ID than normal core load/stores.

61.3.1 DTS register access

A summary of accesses to all DTS registers by bus masters is provided in the following table. Only proper 32-bit accesses are valid. The effects of write accesses that are not 32 bits are not defined.

Table 61-1. DTS register access effects

Register	32-bit read				32-bit write			
	RWA ¹	Core	eDMA	Other crossbar masters	RWA ¹	Core	eDMA	Other crossbar masters
DTS_ENABLE	Data	Data	Data	Data	Data	No effect	No effect	No effect
DTS_STARTUP	Data	Data	Data	Data	Data	No effect	No effect	No effect
DTS_SEMAPHORE	Data and Clear ²	Data	Data	Data	No effect	Bit OR	Bit OR	No effect
DTS_SEMAPHORE_B	Data and Clear ²	Data	Data	Data	No effect	Bit OR	Bit OR	No effect

1. Nexus Read/Write access via an external tool

2. A read of the semaphore registers by Nexus Read/Write Access module is destructive and clears all bits in the register

Access to DTS module registers is controlled based on the XBAR Master ID of the accessing module.

Note

The XBAR Master ID should not be confused with the Master Port number of the XBAR. See the Crossbar Switch (XBAR) chapter for details.

Tools must access the DTS registers through the Nexus Read/Write Access mechanism of a core. External tool accesses through either core appear as if the access is via the core and therefore do not have the same level of access as a Nexus Read/Write Access.

61.4 Memory mapped registers

Only certain types of accesses are allowed to the DTS registers. See the description for each register to determine the accesses that are allowed.

DTS memory map

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
0	Output Enable Register (DTS_ENABLE)	32	R/W	0000_0000h	61.4.1/3057
4	Startup Register (DTS_STARTUP)	32	R/W	0000_0000h	61.4.2/3058
8	Semaphore Register (DTS_SEMAPHORE)	32	R/W	FFFF_FFFFh	61.4.3/3059
C	Semaphore Extension (DTS_SEMAPHORE_B)	32	R/W	FFFF_FFFFh	61.4.4/3060

61.4.1 Output Enable Register (DTS_ENABLE)

The DTS_ENABLE register controls the DTS Trigger Output ($\overline{D\overline{T}O}$) and whether $\overline{D\overline{T}O}$ is active on the $\overline{E\overline{V}T\overline{O}}$ output pin of the device. The DTS_ENABLE register can be read by the core, but can only be written by a Nexus Read Write Access (RWA).

NOTE

Access to the DTS_SEMAPHORE, DTS_SEMAPHORE_B, and DTS_STARTUP registers are unaffected by the state of this register.

Address: 0h base + 0h offset = 0h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	Reserved															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Memory mapped registers

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	Reserved														DTS_EN_B	DTS_EN
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

DTS_ENABLE field descriptions

Field	Description
0–29 Reserved	This field is reserved.
30 DTS_EN_B	DTS Enable B. Controls whether the $\overline{D}\overline{T}\overline{O}$ signal is routed to the $\overline{E}\overline{V}\overline{T}\overline{O}$ pin. The DTS Enable bit is cleared by a device reset (either the assertion of the external $\overline{R}\overline{E}\overline{S}\overline{E}\overline{T}$ or by an internally generated reset). A Nexus reset does not change the state of this register. 0 DTS output disabled. Any bit set in the DTS_SEMAPHORE_B register does not assert the DTS trigger output signal. 1 DTS output enabled. Any bit set in the DTS_SEMAPHORE_B register asserts the DTS trigger output signal ($\overline{D}\overline{T}\overline{O}$).
31 DTS_EN	DTS Enable. Controls whether the $\overline{D}\overline{T}\overline{O}$ signal is routed to the $\overline{E}\overline{V}\overline{T}\overline{O}$ pin. The DTS Enable bit is cleared by a device reset (either the assertion of the external $\overline{R}\overline{E}\overline{S}\overline{E}\overline{T}$ or by an internally generated reset). A Nexus reset does not change the state of this register. 0 DTS output disabled. Any bit set in the DTS_SEMAPHORE register does not assert the DTS trigger output signal. 1 DTS output enabled. Any bit set in the DTS_SEMAPHORE register asserts the DTS trigger output signal ($\overline{D}\overline{T}\overline{O}$).

61.4.2 Startup Register (DTS_STARTUP)

The DTS_STARTUP register is used for tool detection and startup information exchange between external data acquisition tool and the embedded controller. DTS_STARTUP register can be read by the core, the eDMA module and Nexus but can only be updated by a Nexus Read Write Access (RWA).

Address: 0h base + 4h offset = 4h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	AD																															
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

DTS_STARTUP field descriptions

Field	Description
0–31 AD	Application Dependent register bits. The bits have no defined meaning to the microcontroller. They are used to by an external tool to pass information (for example, application options and status) to application

DTS_STARTUP field descriptions (continued)

Field	Description
	software running on target microcontroller at startup time. Use a Nexus RWA 32-bit write access to update the contents of this register. A device reset (either from the $\overline{\text{RESET}}$ pin or an internally generated reset) clears all bits in the register. A Nexus reset does not change the contents of the register.

61.4.3 Semaphore Register (DTS_SEMAPHORE)

The DTS_SEMAPHORE register is used by software to assert the $\overline{\text{DTO}}$ signal on the device $\overline{\text{EVTO}}$ pin. A one in any bit of this register causes the $\overline{\text{DTO}}$ signal on the $\overline{\text{EVTO}}$ pin to be driven low. The intended use of this register is for the $\overline{\text{DTO}}$ signal to notify tools that data is available. Individual bits are used to identify the specific data.

Address: 0h base + 8h offset = 8h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R																																
W	ST																															
Reset	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	

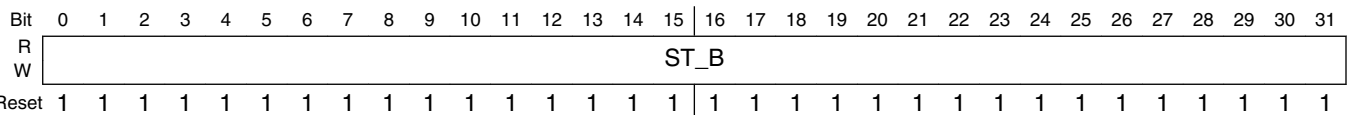
DTS_SEMAPHORE field descriptions

Field	Description
0–31 ST	Semaphore Trigger. When a core or eDMA writes a logical '1' to a bit, the bit is set. A write of '0' by the core or DMA does not change the state of the bit. - All register bits are set to '1' by a device reset. - A Nexus reset does not change the state of this register. - The register can be accessed, with restrictions, by any core, DMA or any Nexus RWA. - For the core or DMA, only 32-bit write or read accesses are valid. - A core or DMA valid read access returns the current value of the register and leaves the register unchanged. 0 No flag 1 Flag is set

61.4.4 Semaphore Extension (DTS_SEMAPHORE_B)

The DTS_SEMAPHORE_B register is used by software to assert the $\overline{DTO}$ signal on the device $\overline{EVTO}$ pin. A one in any bit of this register causes the $\overline{DTO}$ signal on the $\overline{EVTO}$ pin to be driven low. The intended use of this register is for the $\overline{DTO}$ signal to notify tools that data is available. Individual bits are used to identify the specific data.

Address: 0h base + Ch offset = Ch



DTS_SEMAPHORE_B field descriptions

Field	Description
0–31 ST_B	Semaphore Trigger Extension. When a core or eDMA writes a logical '1' to a bit, the bit is set. A write of '0' by the core or DMA does not change the state of the bit. - All register bits are set to '1' by a device reset. - A Nexus reset does not change the state of this register. - The register can be accessed, with restrictions, by any core, DMA or any Nexus RWA. - For the core or DMA, only 32-bit write or read accesses are valid. - A core or DMA valid read access returns the current value of the register and leaves the register unchanged. 0 No flag 1 Flag is set

61.5 Example application

The calibration process of a new engine requires real-time access to calibration tables and the ability to update the tables in real-time The DTS module enables this capability by enabling software to assert a signal to an external device pin to notify an external tool that data is available. The tool can then retrieve the data.

In this type of application the DTS_SEMAPHORE register and DTS Trigger Output (DTO) signal provide a mechanism to notify the calibration tool that the calibration variable or variables (or sets of measurements), up to 32, have been updated with new values and are available for the tool to access.

Note

It is the user's responsibility to ensure that the tool has time to retrieve the data prior to that particular trigger being set a second time. It is also permissible to have multiple triggers active at the same time or for a second trigger to be set before a previous trigger has been serviced, as long as it is not the same trigger (unless it is acceptable to the tool to not receive every data set).

The following table shows an example DTS startup sequence for an external real-time data acquisition system. The startup and synchronization sequence can be as simple or as complicated as the need requires. However, a typical startup sequence is as follows:

1. The DTS_STARTUP register is cleared by a power on reset or any CPU reset.
2. The tool writes a non-zero value to the DTS_STARTUP register.
3. The CPU (user application software) then reads the value of the DTS_STARTUP register. Based on this value, different initialization options can be selected. The bits can be used for any application specific definitions.
4. Since the DTS_SEMAPHORE register is cleared when the tool reads the current value. The tool should perform all necessary initialization before reading this register. The application software can then check that the DTS_SEMAPHORE register was cleared by the tool, to determine that it is safe to start using it for its intended raster trigger semaphore function.
5. An optional hand shake from the CPU can be used to inform the tool that the user software has detected that the tool is attached and the CPU has performed the proper initialization for the tool by writing a predefined value to the DTS_SEMAPHORE register (the example shown in the figure above uses 0xAAAA_AAAA—all A's was used since it is unrealistic that 16 channels could be enabled very quickly after start up after a reset).

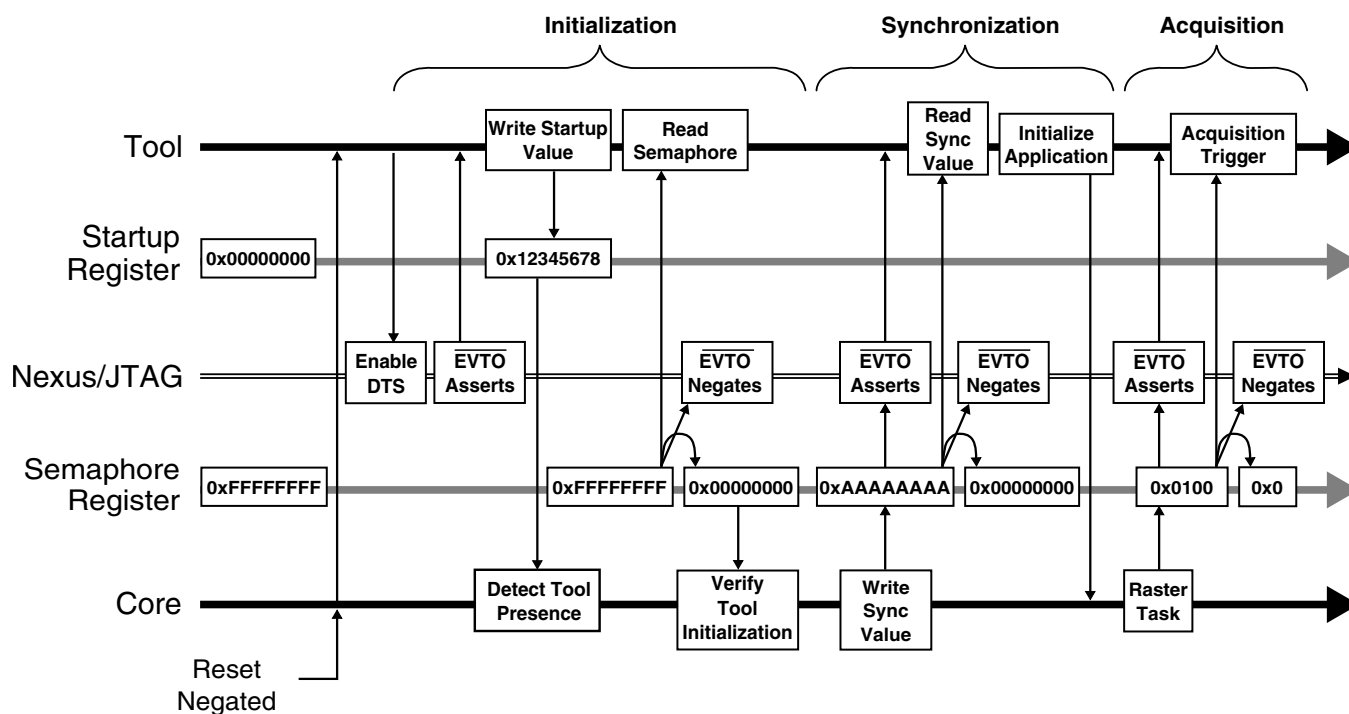


Figure 61-8. DTS startup sequence example

Chapter 62

Fault Collection and Control Unit (FCCU)

62.1 Chip-specific FCCU information

The FCCU on this chip supports 46 non-critical faults (NCFs).

62.1.1 FCCU register default values

The following table documents FCCU register default values that are specific to this chip. For the default values of all other FCCU registers, see [Register descriptions](#).

Table 62-1. FCCU register default values

Register	Default value
FCCU_CFG	0000_0000h ¹
FCCU_NCF_CFG0	FFFF_FFFFh
FCCU_NCF_CFG1	0000_FFFFh
FCCU_NCFS_CFG0	2200_2000h
FCCU_NCFS_CFG1	0000_0000h
FCCU_NCFS_CFG2	0000_0000h
FCCU_NCF_E0	0000_5040h
FCCU_NCF_E1	0000_0000h
FCCU_NCF_TOE0	0000_0000h
FCCU_NCF_TOE1	0000_0000h
FCCU_NCF_TO	0000_FFFFh
FCCU_CFG_TO	0000_0006h

1. FCCU_CFG resets to 0000_0000h after the chip resets the FCCU. After a Configuration-state timeout, the value of FCCU_CFG is 0000_0008h. For details, including the value of FCCU_CFG after an OP31 operation, see [FCCU_CFG register event bit values by source \(N and C\)](#).

62.1.2 FOSU_COUNT value

The FCCU Output Supervision Unit (FOSU) has a timer whose duration is the value FOSU_COUNT. On this chip, the value is 8000 IRCOSC clock cycles.

62.1.3 FCCU fault inputs

The following table identifies the various fault input signals to the FCCU, how they can be tested, and the default reaction to each input after the FCCU itself is reset.

Table 62-2. FCCU non-critical fault (NCF) mapping

NCF	Fault	Fault description	Error reaction path check ¹	Default reaction after reset
0	TEMP_ERROR	Event indication from temperature sensor. The fault is cleared by clearing the status in PMC_ESR_TD, followed by clearing the FCCU channel status, FCCU_NCF_Sn[NCFSm].	LBIST + TSNS_FAULT bit of Temperature Sensor Fault Injection Register (PMC_TS_FIR)	No reaction
1	LVD_ERROR	Voltage out of range indication from LVDs. The fault is cleared by clearing the status in PMC_LVD_HVD_EVENT_STATUS, followed by clearing the FCCU channel status, FCCU_NCF_Sn[NCFSm].	LVD assertion test by tap point manipulation + LVD_FAULT bit of FCCU Fault Injection Register (PMC_FIR)	No reaction
2	HVD_ERROR	Voltage out of range indication from HVDs. The fault is cleared by clearing the status in PMC_LVD_HVD_EVENT_STATUS, followed by clearing the FCCU channel status, FCCU_NCF_Sn[NCFSm].	HVD assertion test by tap point manipulation + HVD_FAULT bit of FCCU Fault Injection Register (PMC_FIR)	No reaction
3	DPMC_SELF_TEST	Digital PMC initialization error during self test. The fault is cleared by clearing the FCCU channel status, FCCU_NCF_Sn[NCFSm]. The status gets cleared if the fault is not persistent.	LBIST+ STEST_FAULT bit of FCCU Fault Injection Register (PMC_FIR)	No reaction
4	MEMORY_DCF_SAFETY_ERR	Indication of fault during timing/repair configuration initialization by SSCM of different memories. Faults can be caused by parity or triple-redundant errors in loaded configuration data. The status is cleared by clearing SSCM_STATUS[CERS], followed by clearing the FCCU channel status, FCCU_NCF_Sn[NCFSm]. The status gets cleared if the fault is not persistent. Additionally, this error will cause a MBIST failure. This channel bit is cleared in case of reset.	LBIST	No reaction

Table continues on the next page...

Table 62-2. FCCU non-critical fault (NCF) mapping (continued)

NCF	Fault	Fault description	Error reaction path check ¹	Default reaction after reset
5	SSCM_XFER_ERR	SSCM transfer or parity error during DCF loading (trimming, STCU, and UTEST Miscellaneous ²). This fault is also asserted when there is a flash memory ECC double bit detection during flash memory reset reads or coherency checks done on the flash memory test row reads. The fault is cleared by clearing SSCM_STATUS[CER], followed by clearing the FCCU channel status, FCCU_NCF_Sn[NCFSm]. This channel bit is cleared in case of reset.	Not testable (but wrong configuration can trigger other detection)	No reaction
6	STCU_UF	STCU unrecoverable error during self-test. Fault is cleared by clearing STCU2_ERR_STAT[UFSF] followed by clearing the FCCU channel status, FCCU_NCF_Sn[NCFSm].	SW check of BIST MISR or Feedback loop via FCCU Fake bit	Reset
7	STCU_RF	STCU recoverable error during self-test. Fault is cleared by clearing STCU2_ERR_STAT[RFSF] followed by clearing the FCCU channel status, FCCU_NCF_Sn[NCFSm].	SW check of BIST MISR or Feedback loop via FCCU Fake bit	No reaction
8	STCU_LMBIST_USR_ERR	Activation of LBIST or MBIST control during application mode. Fault is cleared by clearing the FCCU channel status, FCCU_NCF_Sn[NCFSm]. The status will clear if the fault is not persistent.	Feedback loop via FCCU Fake bit.	No reaction
9	SPURIOUS_DEBUG_SSCM_ACTIVATION	JTAG, NPC, or Debug functionality moved out of reset or SSCM activation during application mode. Fault is cleared by clearing the FCCU channel status, FCCU_NCF_Sn[NCFSm].	LBIST + Feedback loop via FCCU Fake bit	No reaction
10	RCCU_0	Safety cores out of sync during lockstep. Fault is cleared by clearing the FCCU channel status, FCCU_NCF_Sn[NCFSm].	LBIST + Feedback loop via FCCU Fake bit connected to additional RCCU input	No reaction
11	RCCU_1	Safety cores out of sync during lockstep. Fault is cleared by clearing the FCCU channel status, FCCU_NCF_Sn[NCFSm].	LBIST + Feedback loop via FCCU Fake bit connected to additional RCCU input	No reaction
12	WDRS_DBRC0	Core 0 Watchdog Timer reset status (TSR[WRS]) and Core 0 Debug Reset Control (DBCR0[RST]). The fault is cleared by clearing FCCU_NCF_Sn[NCFSm] after the status has been cleared in TSR[WRS].	LBIST + SW test by not triggering WD during startup	Reset
13	WDRS_DBRC1	Core 1 Watchdog Timer reset status (TSR[WRS]) and Core 1 Debug Reset Control (DBCR0[RST]). The fault is cleared by clearing FCCU_NCF_Sn[NCFSm] after the status has been cleared in TSR[WRS].	LBIST + SW test by not triggering WD during startup	No reaction
14	SWTRS0	Software Watchdog A reset. The fault is cleared by clearing FCCU_NCF_Sn[NCFSm] after the status has been cleared inside SWT_A.	LBIST + SW test by not triggering SWT during startup	Reset
15	SWTRS1	Software Watchdog B reset. The fault is cleared by clearing FCCU_NCF_Sn[NCFSm] after the status has been cleared inside SWT_B.	LBIST + SW test by not triggering SWT during startup	No reaction
16	SWTINT0	Software Watchdog A interrupt request. The fault is cleared by clearing FCCU_NCF_Sn[NCFSm] after the status has been cleared inside SWT_A.	LBIST + SW test by not triggering SWT during startup	No reaction

Table continues on the next page...

Table 62-2. FCCU non-critical fault (NCF) mapping (continued)

NCF	Fault	Fault description	Error reaction path check ¹	Default reaction after reset
17	FLASH_INIT ³	Flash memory initiation error. This fault is intended for debug purposes only because, if such an error occurs, no code will run from flash memory. It is expected that this fault is read through development tools.	LBIST	No reaction
18	SSCM_INIT	SSCM initiation error. DCFs might not have loaded properly.	LBIST	No reaction
19	PER_CE	Indication of correctable errors in eTPU, M_CAN, and FlexCAN RAM Controllers. The fault is cleared by clearing the FCCU channel status, FCCU_NCF_Sn[NCFSm]. For proper eTPU SCM and SDM monitoring, eTPU_MESR_MECR[MECIE] must be set to 1.	Error injection in eTPU, FlexCAN, M_CAN + LBIST + Feedback loop via FCCU Fake bit	No reaction
20	PER_UCE	Indication of uncorrectable errors in eTPU, M_CAN, and FlexCAN RAM Controllers. The fault is cleared by clearing the FCCU channel status, FCCU_NCF_Sn[NCFSm]. The eTPU combines uncorrectable errors and other sources in a single exception signal. The exact exception source must be determined with the help of the eTPU Global Exception.	Error injection in eTPU, FlexCAN, M_CAN + LBIST + Feedback loop via FCCU Fake bit	No reaction
21	FLS_CE	Indication of correctable errors in flash memory. The fault is cleared by clearing the FCCU channel status, FCCU_NCF_Sn[NCFSm].	LBIST	No reaction
22	FLS_UCE	Indication of uncorrectable errors in flash memory. The fault is cleared by clearing the FCCU channel status, FCCU_NCF_Sn[NCFSm].	LBIST	No reaction
23	PLL_0	PLL0 loss of lock. For this fault source to be asserted, the Loss of Lock Interrupt must be enabled by writing 1 to PLLDIG_PLL0CR[LOLIE]. Fault is cleared by clearing PLLDIG_PLL0SR[LOLF], followed by clearing FCCU_NCF_Sn[NCFSm]. NOTE: Always configure this fault to generate a reset instead of an interrupt.	LBIST + Forced loss by PLL reconfiguration	No reaction
24	PLL_1	PLL1 loss of lock. For this fault source to be asserted, the Loss of Lock Interrupt must be enabled by writing 1 to PLLDIG_PLL1CR[LOLIE]. Fault is cleared by clearing PLLDIG_PLL1SR[LOLF], followed by clearing FCCU_NCF_Sn[NCFSm]. NOTE: When PLL1 is used as a system clock: Configure this fault to generate a reset instead of an interrupt.	LBIST + Forced loss by PLL reconfiguration	No reaction
25	CMU_0_OSC	Loss of XOSC clock. The fault is cleared by clearing CMU_ISR[OLRI] of CMU_0, followed by clearing the FCCU channel status, FCCU_NCF_Sn[NCFSm]. NOTE: When XOSC is used as a system clock: Configure this fault to generate a reset instead of an interrupt.	LBIST	No reaction
26	CMU_0_PLL	Exception from monitor CMU_0 (PLL0). The fault is cleared by clearing CMU_ISR[FHHI], CMU_ISR[FLLI], or both of CMU_0, followed by clearing the FCCU channel status, FCCU_NCF_Sn[NCFSm].	LBIST + Test by misconfiguration	No reaction

Table continues on the next page...

Table 62-2. FCCU non-critical fault (NCF) mapping (continued)

NCF	Fault	Fault description	Error reaction path check ¹	Default reaction after reset
27	CMU_COMP	Exception from monitors CMU_1, CMU_2, and CMU_3. The fault is cleared by clearing CMU_ISR[FHHI], CMU_ISR[FLLI], or both of CMU_n, followed by clearing the FCCU channel status, FCCU_NCF_Sn[NCFSm].	LBIST + Test by misconfiguration	No reaction
28	CMU_OTHER	Exception from monitors CMU_4 to CMU_8. The fault is cleared by clearing CMU_ISR[FHHI], CMU_ISR[FLLI], or both of CMU_n, followed by clearing the FCCU channel status, FCCU_NCF_Sn[NCFSm].	LBIST + Test by misconfiguration	No reaction
29	EDC_ECC_FLASH	Indication of a hardware fault in the ECC of the flash memory resulting in corrupted ECC detection/correction. Fault is cleared by a dummy read of the flash memory followed by clearing the FCCU channel status, FCCU_NCF_Sn[NCFSm].	LBIST	No reaction
30	EDC_ECC_FLASH_C	Indication of hardware fault in the Flash Memory Controller resulting in corrupted ECC detection/correction. Fault is cleared by clearing the FCCU channel status, FCCU_NCF_Sn[NCFSm].	LBIST	No reaction
31	ENC_ERR_FLASH	Indication of hardware fault resulting in corrupted flash memory access. Fault is cleared by clearing the FCCU channel status, FCCU_NCF_Sn[NCFSm].	LBIST	No reaction
32	ADDR_FDBK_ERROR_FLASH_C	Indication of an addressing error in the Flash Memory Controller resulting in corrupted flash memory access. Fault is cleared by clearing the FCCU channel status, FCCU_NCF_Sn[NCFSm].	LBIST	No reaction
33	XBAR Transaction Monitor	Indication by the Crossbar Integrity Checker (XBIC) of a corrupted transaction in the crossbar. Fault is cleared by clearing the FCCU channel status, FCCU_NCF_Sn[NCFSm].	LBIST + proper configuration of XBIC Error Injection Register (XBIC_EIR)	No reaction
34	ADDR_FDBK_ERROR_PRAM + RAM_LWB_ERROR ⁴	Indication of an addressing error in system RAM or a write error in the RAM Controller resulting in corrupted RAM access. Fault is cleared by clearing the FCCU channel status, FCCU_NCF_Sn[NCFSm].	LBIST	No reaction
35	SSCM_PASS_ERROR	An ECC error occurred while SSCM or PASS was fetching data from flash memory.	LBIST	No reaction
36	EDC_ERROR_PRAM	Indication of a fault in the RMW logic of the system RAM resulting in an incorrect RMW access and a potentially corrupted RAM word. Fault is cleared by clearing the FCCU channel status, FCCU_NCF_Sn[NCFSm].	LBIST	No reaction
37	DFT1	Test circuitry Group 1 activation. Fault is cleared by clearing the FCCU channel status, FCCU_NCF_Sn[NCFSm].	LBIST	No reaction
38	DFT2	Test circuitry Group 2 activation. Fault is cleared by clearing the FCCU channel status, FCCU_NCF_Sn[NCFSm].	LBIST	No reaction
39	DFT3	Test circuitry Group 3 activation. Fault is cleared by clearing the FCCU channel status, FCCU_NCF_Sn[NCFSm].	LBIST	No reaction
40	DFT4	Test circuitry Group 4 activation. Fault is cleared by clearing the FCCU channel status, FCCU_NCF_Sn[NCFSm].	LBIST	No reaction

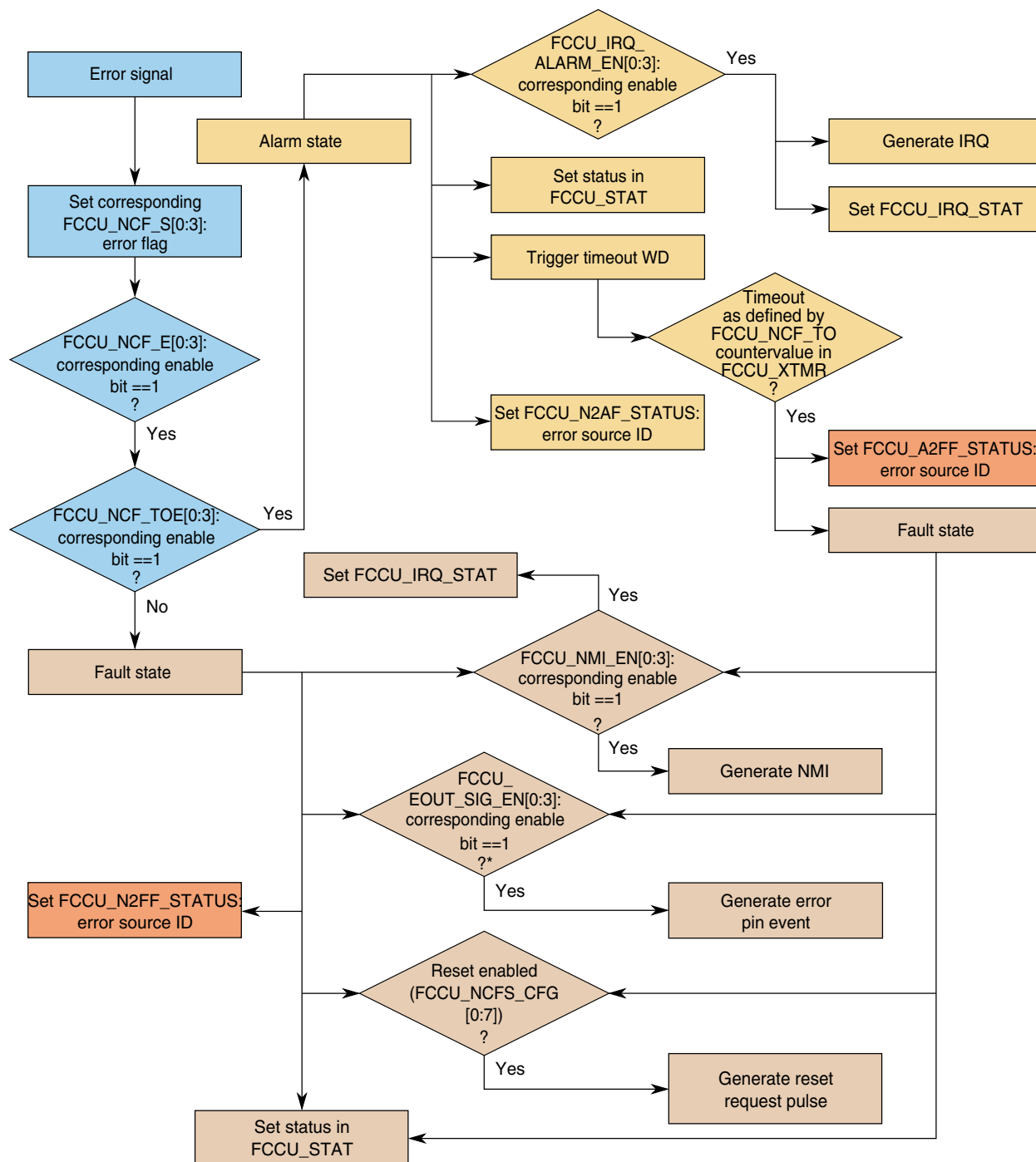
Table continues on the next page...

Table 62-2. FCCU non-critical fault (NCF) mapping (continued)

NCF	Fault	Fault description	Error reaction path check ¹	Default reaction after reset
41	Safety Core Machine Check Exception	Safety Core exception indication. The fault is cleared by clearing the status in MCSR register of the Safety core followed by clearing the FCCU channel status, FCCU_NCF_Sn[NCFSm].	LBIST + willfully cause an exception (for example, access a non-existing address)	No reaction
42	Lockstep mode	Indication of disabled Checker Core or RCCUs. Fault is cleared by clearing the FCCU channel status, FCCU_NCF_Sn[NCFSm].	LBIST	No reaction
43	ERROR_IN_ERR	Error input pin (external to the MCU) signaling an error condition. The fault is cleared by clearing the fault driving the pin followed by clearing the FCCU channel status, FCCU_NCF_Sn[NCFSm].	External activation of EIN. External entity must pull the error input low.	No reaction
44	Unused	—	—	—
45	ERM_CE	ERM module detected single ECC error. The fault is cleared by clearing the FCCU channel status, FCCU_NCF_Sn[NCFSm].	EIM + LBIST + Feedback loop via FCCU Fake bit	No reaction
46	ERM_UCE	ERM module detected multiple ECC error. The fault is cleared by clearing the FCCU channel status, FCCU_NCF_Sn[NCFSm].	EIM + LBIST + Feedback loop via FCCU Fake bit	No reaction

1. All channels can use feedback loop via FCCU Fake bit.
2. TDM does not have parity error during DCF loading. For UTEST Miscellaneous, only the DCF clients for PMC Digital have parity detection.
3. This fault occurs after any flash memory programming. To avoid unexpected behavior, configure the reaction to this fault carefully if flash memory programming is executed.
4. Fault 34 might be asserted during MBIST and LBIST execution. Before running MBISTs, configure this channel to result in no reset reaction. If this channel becomes asserted during MBIST testing, simply clear the fault and take no further action.

62.1.4 Error signal flow diagram



* This condition must be satisfied only when FCCU is configured for Bistable fault-output mode (FCCU_CFG[FOM]).

Figure 62-1. Error signal flow

62.1.5 Restriction on configuring RESET and NMI reactions on same FCCU channel

Do not configure an FCCU channel to have both a RESET reaction (using the FCCU_NCFS_CFG n register) and an NMI reaction (using the FCCU_NMI_EN n register) at the same time. If the FCCU channel is serviced and the chip resets, the NMI interrupt address is undefined and halts the chip.

62.1.6 FCCU_CFG register event bit values by source (N and C)

For each bit in the [Configuration \(FCCU_CFG\)](#) register, this table specifies the values of N and C for this chip. For information on how to interpret and use these values, see [FCCU_CFG register bit value sources \(N and C\) by event](#).

Bit	N	C
0	0	0
1	0	0
2	0	0
3	0	0
4	0	0
5	0	0
6	0	0
7	—	0
8	—	0
9	—	0
10	—	0
11	—	0
12	—	0
13	—	0
14	—	0
15	—	0
16	0	0
17	0	0
18	0	0
19	—	0
20	0	0
21	0	0
22	0	0
23	0	0
24	0	0

Table continues on the next page...

Bit	N	C
25	0	0
26	0	0
27	0	0
28	0	1
29	0	0
30	0	0
31	0	0

62.2 Introduction

The Fault Collection and Control Unit (FCCU) offers a hardware channel to collect faults and to place the device into a safe state when a failure in the device is detected. No CPU intervention is requested for collection and control operation.

The FCCU offers a systematic approach to fault collection and control. The distinctive features of the module are:

- Redundant collection of hardware checker (for example, RCCU) results
- Redundant collection of fault information from safety relevant modules on the device
- Collection of test results
- Configurable fault control
- Configurable internal reactions for each NCF:
 - No reaction
 - IRQ
 - Functional reset
 - NMI
- External reactions via configurable output pins
- Watchdog timer for the reconfiguration phase
- Lockable configuration
- Nonvolatile memory interface (NVMCFG signals) for configuration loading
- One of the fault-output (EOUT) signals is high to indicate operational (OK) state (in bi-stable operation mode). The above is not true in case the error out protocol is configured to be a toggling protocol, for example, dual rail and time switching.

- After power on, the EOUT signals have high impedance.¹ They show operational state only after configuration by software.
- In case of a failure event or on software request for Error pin indication, the pin(s) are set to faulty state for a minimum time T_{min} , even if SW tries to release it before (for the case of error pin configured in Bi-stable mode only).
- The overall fault detection, processing and indication time is less than 10 ms.
- The actual maximum time is 5 CLKSAFE cycles.

The FCCU circuitry is checked at the start-up by the self-checking procedure. The FCCU is operative with the default configuration immediately after the completion of the self-checking procedure. Internal (functional reset request pulse, interrupt request) and external (EOUT signaling) reactions are statically defined or programmable. The default configuration can be modified only in the CONFIG state. FCCU is designed to function when CLKSYS is faster than the CLKSAFE clocks.

62.2.1 Acronyms and abbreviations

Term	Description
CF	Critical fault
CPU	Central processing unit
EOUT	Error out
FOSU	FCCU output supervision unit
FSM	Finite state machine
INTC	Interrupt controller
intf	Interface
IRQ	Interrupt request
NCF	Noncritical fault
NMI	Nonmaskable interrupt
NVM	Nonvolatile memory
RCC	Redundancy control checkers
RCCU	Redundancy control checker unit
SoC	System-on-chip
STCU	Self-test control unit
SW	Software

1. Actual value depends on device specific setting at pad level.

62.3 Main features

- Management of noncritical faults
- HW or SW fault recovery management
- Fault detection and collection
- Fault injection (fake faults)
- External reaction (fault state): EOUT signaling. Error indication via the pin(s) is controlled by the FCCU.
- Internal chip reactions (alarm state): interrupt request
- Internal chip reactions (fault state):
 - functional reset
 - NMI
- Bi-Stable, Dual-Rail and Time Switching output protocols on EOUT
- Internal (to the FCCU) watchdog timer for the reconfiguration phase
- Configuration lock
- Nonvolatile memory interface (NVMCFG signals) for configuration loading

62.4 Block diagram

The following figure is a top-level diagram of the FCCU module.

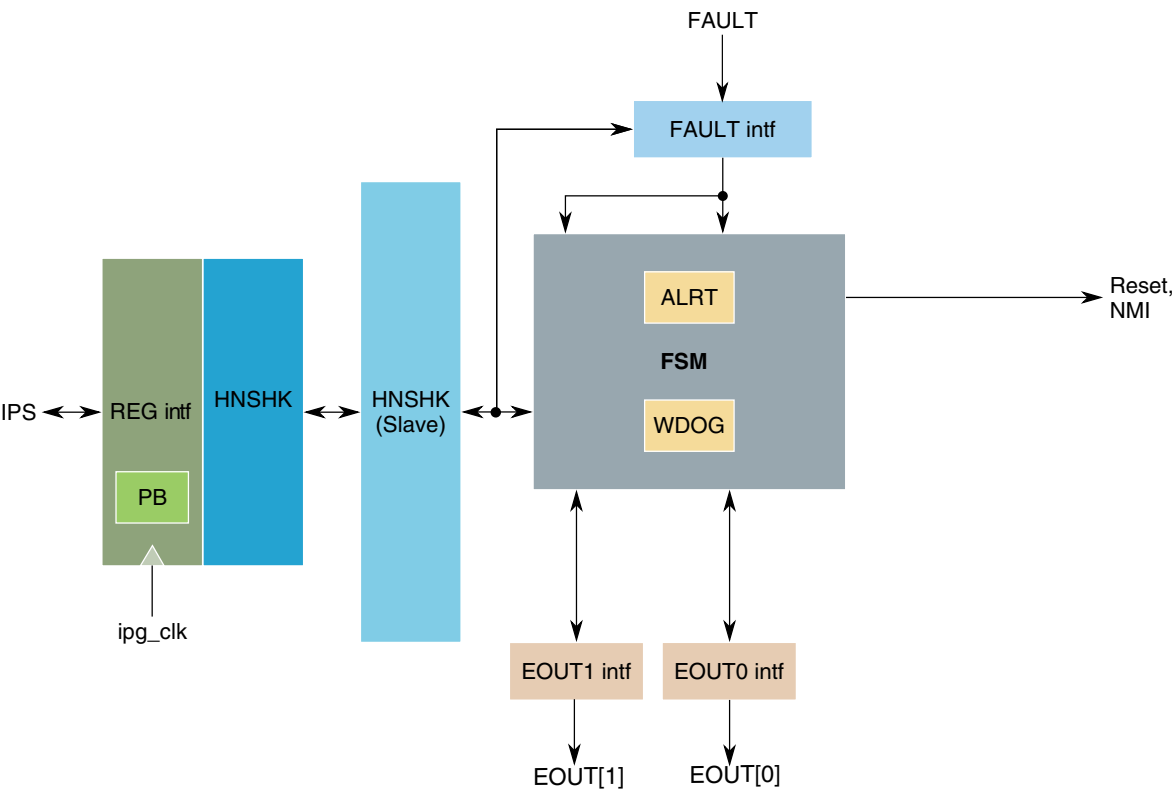


Figure 62-2. FCCU block diagram

This table describes the FCCU submodules.

Table 62-3. FCCU submodules

Submodule	Description
REG intf	Includes the register file, the IPS bus interface, the IRQ interface and the parity block (PB) for the configuration registers
HNSHK blocks (master and slave blocks)	Includes the FSM's ability to support the handshake between the REG interface and the FSM unit due to the usage of 2 asynchronous clocks (IPS system clock and RC oscillator clock)
FSM unit	Implements the main functions of the FCCU. The FSM also includes the: • Watchdog timer (WDG) • Alarm timer (ALRT)
FAULT intf	Implements the interface for the fault conditioning and management
EOUTx units	Implement the output stage to manage the EOUT interfaces

62.5 Signal description

62.5.1 Signals

This table describes the signals on the boundary of FCCU.

Signal	Direction	Function	Active level	Clock
Clock and reset				
CLKSYS	Input	System Clock—Supplies the primary clock.	Rising edge	—
CLKSAFE0	Input	Safe Clock 0—Supplies one of two 16 MHz clocks for use by various redundant submodules. This clock and CLKSAFE1 are often referred to as the same clock (CLKSAFE).	Rising edge	—
CLKSAFE1	Input	Safe Clock 1—Supplies one of two 16 MHz clocks for use by various redundant submodules. This clock and CLKSAFE0 are often referred to as the same clock (CLKSAFE).	Rising edge	—
EOUT interface				
EIN[1:0]	Input	Error Input—Provide the mechanism for FCCU to capture the states that off-chip logic drives on the associated EOUT[1:0] signals. (On the chip, the EOUT signals are connected to their corresponding EIN signals.) FCCU captures the states in the EIN1 and EIN0 fields.	—	Asynchronous
EOUT[1:0]	Output	Error Output—Indicate FCCU's condition (Faulty, Nonfaulty, or Configuration) to off-chip logic.	Programmable	CLKSAFE0, CLKSAFE1
FOSU interface				

Table continues on the next page...

Signal description

Signal	Direction	Function	Active level	Clock
FIF	Output	FCCU In Fault—Indicates to the FOSU module, if present, that FCCU is recognizing a fault. FCCU asserts and then deasserts this signal in two cases: <i>Case 1</i> When the EOUT signals are programmed to be always low (FCCU_SET_CLEAR), FCCU asserts this signal until the EOUT signals are programmed not to be always low. <i>Case 2</i> When the EOUT signals are programmed not to be always low (FCCU_SET_CLEAR), FCCU asserts this signal when all of the following are true: - FCCU enters Fault state as the result of a fault. - If the fault is a noncritical fault, and the EOUT signals are in a fault-output mode (FOM) other than Bistable, the associated EOUTENx field is set to enabled. FCCU deasserts this signal when FCCU leaves Fault state.	High	CLKSAFE0, CLKSAFE1
Interrupt interface				
NMIOUT	Output	Nonmaskable Interrupt Output—Sends a nonmaskable interrupt request to the processor core or cores when FCCU enters Fault state as the result of a noncritical fault for which NMI is enabled as the reaction (FCCU_NMI_ENa[NMIENx]).	Low	CLKSYS
NVM interface				
NVMCFG[21:0]	Input	Nonvolatile Memory Configuration—Provide the values of certain bits in the Configuration (FCCU_CFG) register when certain events occur. See Configuration (FCCU_CFG). Some or all of these signals may be driven by bits in a data record in nonvolatile (flash) memory or may be tied high (1) or low (0). See FCCU_CFG register bit value sources (N and C) by event.	Static value	—

Table continues on the next page...

Signal	Direction	Function	Active level	Clock
SSCMDONE	Input	SSCM Done—After an FCCU reset, indicates that the SSCM module is now driving any NVMCFG signals that are associated with a data record stored in nonvolatile (flash) memory.	High	CLKSYS

62.6 Put FCCU in Configuration or Normal state

62.6.1 Introduction

You put FCCU in Configuration state to change its configuration. You put FCCU in Normal state to save changes to the configuration.

62.6.2 About changing states

When changing states, keep the following in mind:

- To put FCCU in Configuration state, FCCU must be in Normal state.
- After FCCU is reset, the configuration is temporarily locked. You must temporarily unlock the configuration before you can put FCCU in Configuration state.
- When FCCU is in Configuration state, FCCU doesn't actually save the changes you make to the configuration. To save changes to the configuration, you must manually put FCCU in Normal state. If FCCU automatically leaves Configuration state and enters Normal state because the configuration-timer interval (**TO**) expires (called a Configuration-state timeout), FCCU changes the value of the [Configuration \(FCCU_CFG\)](#) register to its Configuration-state-timeout value and the value of each of the other configuration registers to its reset value. For information on the Configuration-state timeout value, see [FCCU_CFG register bit value sources \(N and C\) by event](#). For a list of configuration registers, see [Configuration registers](#).

62.6.3 Put FCCU in Configuration state

1. Temporarily unlock the configuration (see [Temporarily unlock the configuration](#)).
2. To determine FCCU's state, run the OP3 operation (see [Run an operation](#)).
3. Check the operation status (**OPS**).
 - If the operation status is Successful, go to step 4.
 - If the operation status is Aborted, go to step 2.

4. Check the FCCU status ([STATUS](#)).
 - If the FCCU status is Configuration, FCCU is in Configuration state. Stop.
 - If the FCCU status is Normal, go to step [5](#).
 - If the FCCU status is Alarm or Fault, go to step [7](#).
5. To put FCCU in Configuration state, run the OP1 operation (see [Run an operation](#)).
6. Check the operation status ([OPS](#)).
 - If the operation status is Successful, FCCU is in Configuration state. Stop.
 - If the operation status is Aborted, the configuration is probably permanently locked. Go to step [9](#).
7. Recover all faults.
8. Go to step [2](#).
9. Reset FCCU.

62.6.4 Put FCCU in Normal state

1. To determine FCCU's state, run the OP3 operation (see [Run an operation](#)).
2. Check the operation status ([OPS](#)).
 - If the operation status is Successful, go to step [3](#).
 - If the operation status is Aborted, go to step [1](#).
3. Check the FCCU status ([STATUS](#)).
 - If the FCCU status is Normal, FCCU is in Normal state. Go to step [8](#).
 - If the FCCU status is Configuration, go to step [4](#).
 - If the FCCU status is Alarm or Fault, go to step [6](#).
4. To put FCCU in Normal state, run the OP2 operation (see [Run an operation](#)).
5. Check the operation status ([OPS](#)).
 - If the operation status is Successful, FCCU is in Normal state. Go to step [8](#).
 - If the operation status is Aborted, go to step [4](#).
6. Recover all faults.
7. Go to step [1](#).
8. If you want to prevent changes to the configuration, lock it:
 - To require a key to unlock the configuration, temporarily lock the configuration (see [Temporarily lock the configuration](#)).
 - To require a reset of FCCU to unlock the configuration, permanently lock the configuration (see [Permanently lock the configuration](#)).

62.7 Run operations

62.7.1 Introduction

You run operations to perform actions such as putting FCCU in Configuration state or reading a timer. For a complete list of operations you can run, see [OPR](#).

62.7.2 About running operations

When running operations, keep the following in mind:

- FCCU ignores any operations initiated while the operation status is In Progress.
- Certain operations must be unlocked before you can initiate them. After you initiate them, they are again locked.

62.7.3 Run an operation

1. Check the operation status ([OPS](#)).
2. If the operation status is In Progress, go to step [1](#).
3. If the operation must be unlocked before you can initiate it, unlock the operation ([CTRLK](#)) using a 32-bit write.

NOTE

The [Control Key \(FCCU_CTRLK\)](#) register used in this step and the [Control \(FCCU_CTRL\)](#) register used in the next step must be written with consecutive instructions. Do not use read-modify-write instructions, such as bit-field instructions, to modify these registers.

4. Initiate the operation ([OPR](#)) using a 32-bit write.
5. Check the operation status ([OPS](#)).
6. If the operation status is In Progress, go to step [5](#).

The operation status is now Idle (OP15 only), Aborted, or Successful.

62.8 Lock and unlock the configuration

62.8.1 Introduction

After you configure FCCU, you can lock its configuration to prevent accidental changes.

62.8.2 About locking the configuration

When locking the configuration, keep the following in mind:

- If you attempt to lock the configuration while FCCU is in Configuration state, FCCU doesn't actually lock the configuration until FCCU leaves Configuration state—that is, either you put FCCU in Normal state, or FCCU puts FCCU in Normal state because the configuration-timer interval ([TO](#)) expires.
- After you permanently lock the configuration, you must reset FCCU before you can put FCCU in Configuration state.

62.8.3 Temporarily lock the configuration

1. From Supervisor mode, write a lock value to [TRANSKEY](#).
2. Put FCCU in Normal state if it's in Configuration state (see [Put FCCU in Normal state](#)).

62.8.4 Temporarily unlock the configuration

- From Supervisor mode, write the unlock value to [TRANSKEY](#).

62.8.5 Permanently lock the configuration

1. From Supervisor mode, write the lock value to [PERMNTKEY](#).
2. Put FCCU in Normal state if it's in Configuration state (see [Put FCCU in Normal state](#)).

The configuration is permanently locked until FCCU is reset.

62.9 Functional description

62.9.1 Definitions

In general, the following definitions are applicable for the fault management:

- **HW recoverable fault:** the fault indication is an edge-triggered and level-sensitive signal that remains asserted until the fault cause is deasserted. That is, if logical 0 on the fault signal indicates fault, then the status flags are valid as long as the fault line stays at 0. The status is automatically cleared when the fault signal goes to 1. Typically the fault signal is latched external to the FCCU in the module where the fault occurred. The FCCU state transitions are consequently executed on the state changes of the input fault signal. No SW intervention in the FCCU is required to recover the fault condition.
- **SW recoverable fault:** the fault indication is a signal asserted without a defined time duration. The fault signal is latched in the FCCU. The fault recovery is executed following a SW recovery procedure (status/flag register clearing).

HW recoverable is an option to exclude the handling of error source/s by FCCU management SW, in case it is known that the fault is recoverable by itself when the fault condition gets corrected.

62.9.2 FSM description

FCCU functionality is depicted by the finite state machine (FSM) state diagram.

Basically four states are identified with the following meaning:

- **Configuration:** Used only to modify the configuration of FCCU from its default. A subset of the FCCU registers, dedicated to define the FCCU configuration (global configuration, reactions to fault, timeout, noncritical fault masking) can be accessed in write mode only in Configuration state.

Configuration state is accessible only in Normal state and if the configuration is not locked. The permanent configuration lock can be disabled by a reset which also resets the FCCU. The transient lock register is unlocked by writing BCh into it. FCCU gets transiently locked again if an invalid key is written into FCCU_TRANS_LOCK register (that is, other than BCh). Key to lock FCCU permanently is FFh, written in the FCCU_PERMNT_LOCK register.

After the release of reset the state of the transient lock will be locked. The state of the permanent lock will be unlocked.

The locking feature only restricts the FSM movement into Configuration state. Once the user enters Configuration state and then tries to lock the configuration, the locking of configuration will be effective after FCCU moves to Normal state, it will not be effective in the current Configuration state.

The Configuration to Normal state transition can be executed by SW or automatically following a timeout condition of the watchdog. In case the timeout information and the SW request to mode change to Normal appears at the same time, watchdog timeout has the priority and hence the Configuration registers (those that are writable only in Configuration state) are reset to their default values. The movement to Normal is made.

The incoming faults, occurring during the configuration phase (Configuration state) are latched in order to process them when FCCU is moved into Normal state, according to the new configuration.

All pending faults that occur during Configuration state result in both of the following:

- Highest-priority state transition
- Interrupt generation (NMI or alarm IRQ)

If the state transition occurs, it gives the reset reaction corresponding to the worst case based on all the faults (pending or nonpending faults) that occurred during Configuration state.

- Normal: This is FCCU's operating state when no faults are occurring. It is also the default state on the reset exit. Following state transitions occur on one of the following events:
 - unmasked noncritical faults with the timeout disabled → FCCU moves to Fault state
 - unmasked noncritical faults with the timeout enabled → FCCU moves to Alarm state
 - masked noncritical faults → FCCU stays in Normal state
- Alarm: FCCU moves into Alarm state when an unmasked noncritical fault occurs and the timeout is enabled. The transition to Alarm state goes along with an interrupt alarm, if enabled. By definition, this fault may be recovered within a programmable timeout period, before it generates a transition to Fault state. The timeout is reinitialized if FCCU enters Normal state. The timeout restarts following the recovery from Fault state.
- Fault: FCCU moves into Fault state when one of the following condition occurs:
 - timeout related to a noncritical fault when FCCU is in Alarm state
 - unmasked noncritical faults with the timeout disabled

The transition from Normal/Alarm to Fault state goes along with the generation of:

- NMI interrupt (optional)
- EOUT signaling (optional)
- SW option: functional reset
- Non Maskable Interrupt (NMI) is routed only to the Safety Core

After moving to Fault state, if there is either a previous pending fault or a new fault for which NMI is enabled, NMI generation takes place.

Multiple faults can occur at the same time. Consider the case where there is a fault for which Alarm state is enabled and IRQ is programmed along with a fault for which Alarm state is disabled. In that case, the FSM moves to Fault state, but with the generation of alarm IRQ for the fault which has IRQ enabled.

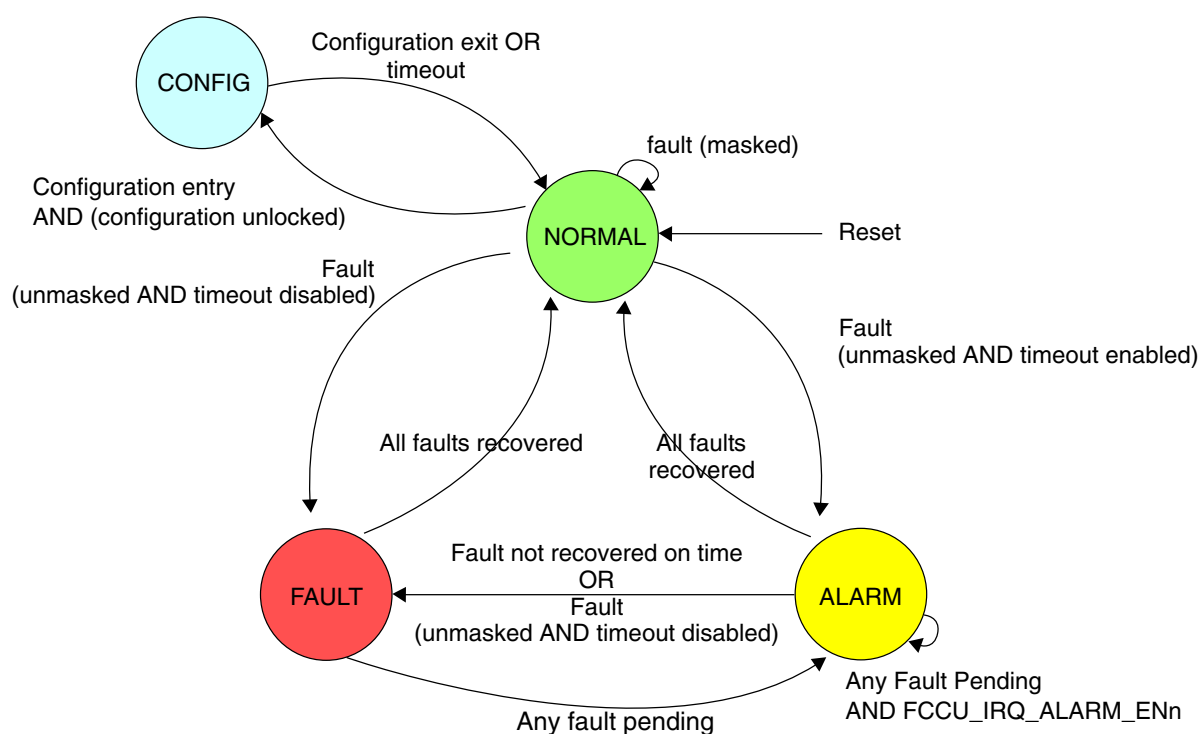


Figure 62-3. FCCU state diagram

62.9.3 Fault priority scheme and nesting

The FAULT state has a higher priority than the ALARM state in case of concurrent fault events (noncritical) that occur in the NORMAL state.

The ALARM to FAULT state transition occurs if a noncritical fault (unmasked and with time-out disabled) is asserted in the ALARM state.

The ALARM to NORMAL state transition occurs only if all the noncritical faults (including the faults that have been collected after the entry in the ALARM state) have been cleared (SW or HW recovery); otherwise the FCCU will remain in the ALARM state.

The FAULT to NORMAL state transition occurs only if all the noncritical faults (including the faults that have been collected after the entry in the FAULT/ALARM state) have been cleared (SW or HW recovery); otherwise the FCCU will move to the ALARM state (if any noncritical fault is still pending and the time-out is not elapsed).

In general, no fault nesting is supported except for the noncritical faults that cause an ALARM to FAULT state transition. In this case the NCF timer is stopped until the FAULT state is recovered. If FCCU is in ALARM state and another fault comes, which has its alarm timeout enabled, then the alarm timer shall not reload and shall not start again.

62.9.4 STCU interface

The STCU interface includes:

- A set of signals resulting from the self-checking procedure connected externally at the FCCU.

The STCU fault signals are processed by the FCCU when the SoC is re-booted following the self-testing procedure. The STCU includes a status register that stores the self-testing results (flags).

- a mask that inhibits the EOUT dummy signaling until the STCU self-checking procedure has been completed.

During the self-testing procedure, depending on the STCU results, three cases are applicable:

1. STCU completes the self-testing procedure successfully. The SoC reboots and the FCCU is responsible for fault collection and indication.
2. STCU completes the self-testing procedure with low severity failures. The FCCU records the fault condition but does not, by default, trigger a reset cycle.
3. STCU completes the self-testing procedure with serious failures. The FCCU triggers a reset cycle in response to this fault input from the STCU. Because the fault condition survives reset, new reset cycles are subsequently triggered until the reset escalation feature takes control, freezing the chip in the reset state until a POR.

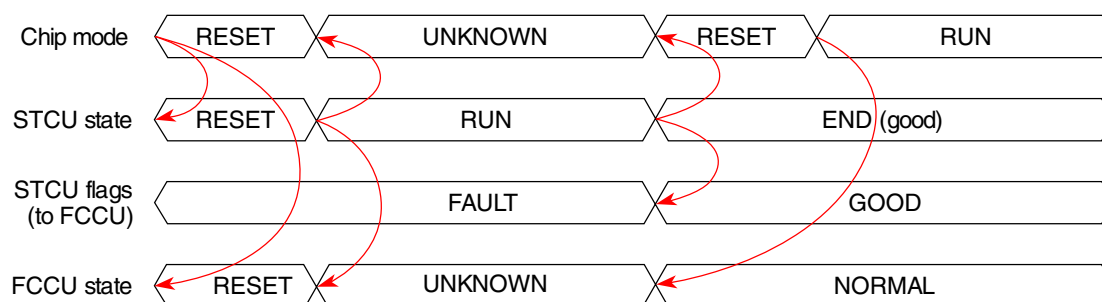


Figure 62-4. STCU-FCCU (case 1)

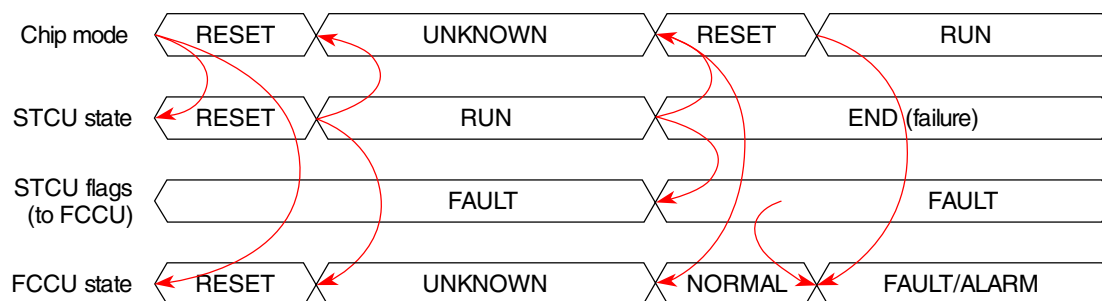


Figure 62-5. STCU-FCCU (case 2)

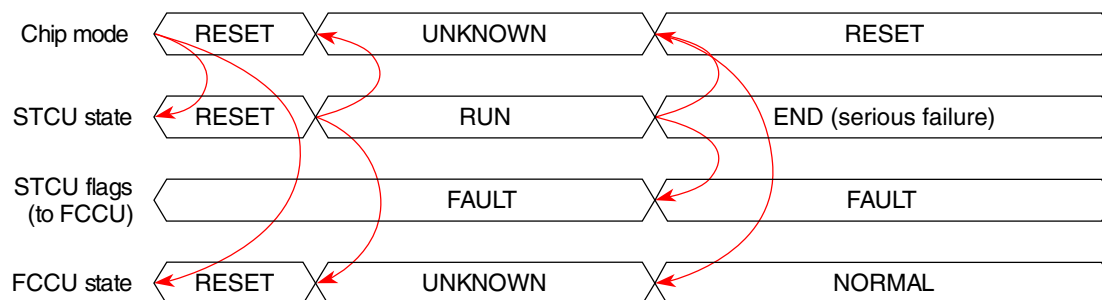


Figure 62-6. STCU-FCCU (case 3)

62.9.5 Nonvolatile memory interface

FCCU includes a nonvolatile memory interface (NVMCFG signals) that provides the values of certain bits in the [Configuration \(FCCU_CFG\)](#) register when certain events occur. See [Configuration \(FCCU_CFG\)](#). Whether any of these signals are actually connected to nonvolatile (flash) memory is chip-specific. Some or all of these signals may be driven by bits in a data record in nonvolatile (flash) memory or may be tied high (1) or low (0). See [FCCU_CFG register bit value sources \(N and C\) by event](#).

The timing diagram related to the NVM interface is given in [Figure 62-7](#).

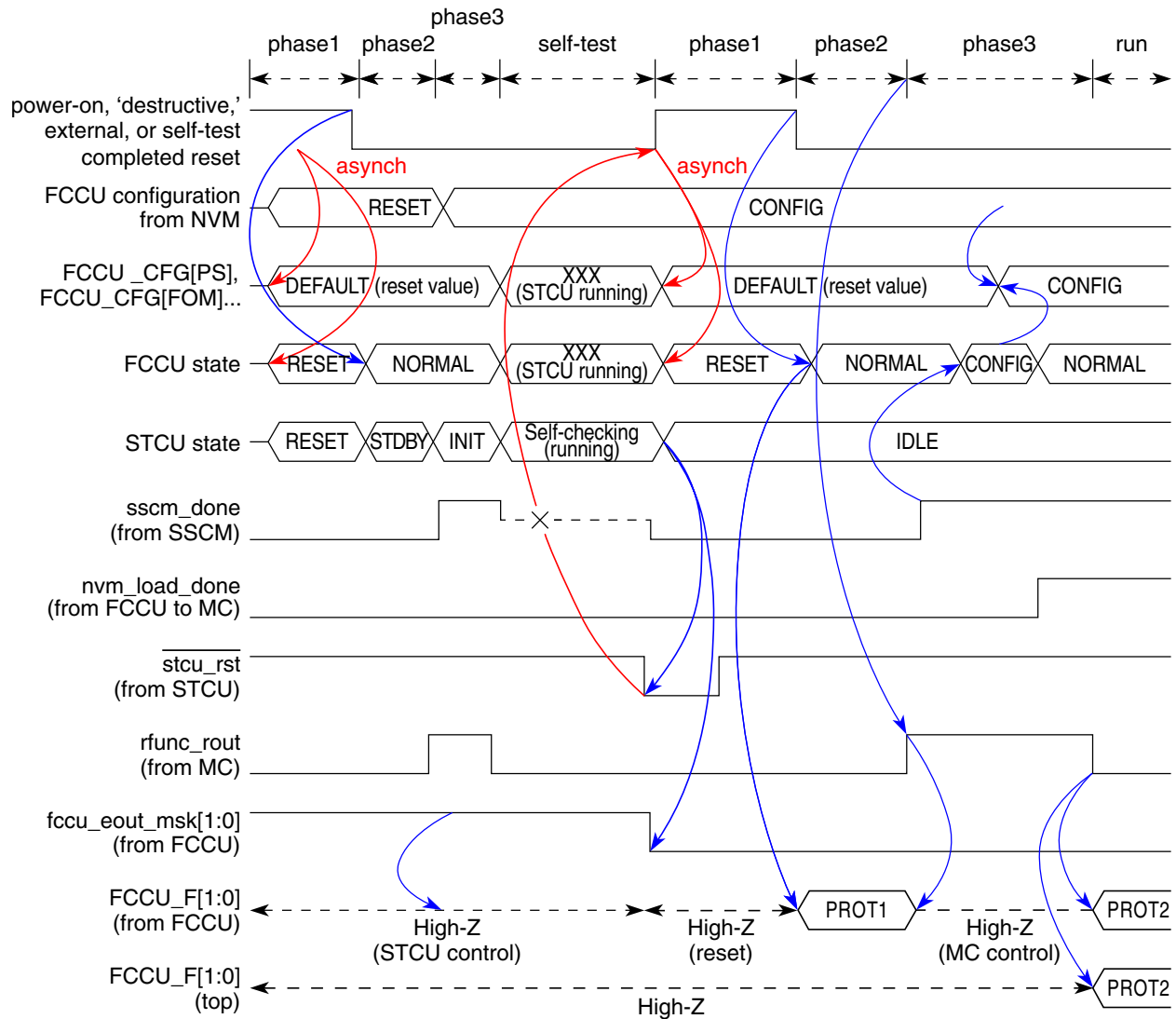


Figure 62-7. NVM interface

62.9.6 EOUT interface

62.9.6.1 Introduction

You use the EOUT[1:0] signals to indicate FCCU's condition to off-chip logic. For information on the availability and names of these FCCU signals on the boundary of this chip, see the chip-specific FCCU information.

62.9.6.2 The FCCU conditions

There are three FCCU conditions:

Condition	Description
Faulty	The fault-output (EOUT) timer is running (see How the fault-output (EOUT) timer works).
Nonfaulty	All of the following are true: • The fault-output (EOUT) timer is not running. • FCCU is in Alarm or Normal state.
Configuration	All of the following are true: • The fault-output (EOUT) timer is not running. • FCCU is in Configuration state.

62.9.6.3 How the fault-output (EOUT) timer works

FCCU starts the fault-output (EOUT) timer when all of the following are true:

- The fault-output (EOUT) signals are not programmed to be always low ([FCCU_SET_CLEAR](#)).
- The fault-output (EOUT) timer isn't already running.
- FCCU enters Fault state as the result of a fault.

When the fault-output (EOUT) timer is already running and a new fault occurs:

- When FCCU is in Configuration state: FCCU doesn't restart the fault-output (EOUT) timer.
- When FCCU is in Normal state or Alarm state:
 - And Alarm state is enabled for the fault (noncritical): FCCU enters (or remains in) Alarm state but doesn't restart the fault-output (EOUT) timer.
 - And Alarm state is disabled for the fault (noncritical): FCCU enters Fault state and restarts the fault-output (EOUT) timer.
- When FCCU is in Fault state: FCCU restarts the fault-output (EOUT) timer.

FCCU stops and reinitializes the fault-output (EOUT) timer when all of the following are true:

- If the EOUT signals are in Bistable fault-output mode ([FOM](#)), T_min has expired.
- All faults that caused FCCU to enter or remain in Fault state since FCCU started the fault-output (EOUT) timer have been cleared, causing FCCU to return to Normal state.

62.9.6.4 Prepare the EOUT signals to indicate FCCU's condition

Make sure all of the following are true:

- The EOUT signals are active ([FCCU_SET_AFTER_RESET](#)).
- The EOUT signals are controlled by FCCU's finite state machine ([FCCU_SET_CLEAR](#)).

62.9.6.5 More about the EOUT interface

Different fault-output modes (protocols) for the fault-output (EOUT) interface are supported (FCCU_CFG[FOM]):

- Dual-Rail
- Time-Switching
- Bistable
- Test

You can further configure the fault-output modes using the following attributes:

Attribute	Field	Setting used in the example diagrams and tables that follow
Switching mode	FCCU_CFG[SM]	Slow
Polarity selection	FCCU_CFG[PS]	For the faulty indication, EOUT1 is high, and EOUT0 is low.

The EOUT frequency is generated by dividing the CLKSAFE frequency by a fixed factor of 2^{18} .

With a CLKSAFE frequency of 16 MHz, this drives a signal of 61 Hz on EOUT.

The external monitor of the EOUT protocol should oversample the EOUT signals in order to synchronize periodically the external clock (used by the monitor) and CLKSAFE (which detects the edge transition of the EOUT protocol) in Dual-Rail or Time-Switching fault-output mode.

In case of a failure event or on software request for EOUT indication, the signal(s) are set to the faulty state for a minimum time T_{min} , even if software tries to release it before. If software configures the error pins to OK(1), and if a fault comes trying to drive the pin to NOK(0), then priority is given to the fault indication and the error signals indicate NOK, such as an incoming fault is not masked when software has set the error signal to high. During the T_{min} by a non-software fault, the FCCU FSM moves independently of this

signal state (low), and as soon as the timer expires, the pin behavior is dictated by the state in which the FSM finds itself in, and it is not possible to set the signals to OK by software moving FCCU to Configuration state, as long as this timer is running. No software intervention is needed to bring the signal from the low state.

Software can bring the pin back to OK state by clearing the faults and waiting for the T_min interval to expire, after which the FCCU automatically enters Normal state and the error signal indicates OK.

In case another failure event happens within T_min after a first one, the T_min counter is restarted.

NOTE

The transition from Fault to Configuration state is not possible but is shown to display the behavior in all four states—reset, normal, error, and configuration.

NOTE

FCCU remains in Normal state when a fault is disabled; therefore, NA (not applicable) appears in the following table.

NOTE

The following table is valid only when the EOUT signals are active ([FCCU_SET_AFTER_RESET](#)).

Table 62-4. FCCU error pin behavior on state transitions

Fault signaling state	Transition from Normal to Fault state			Normal+Configuration states			Transition from Normal to Alarm state		
	with Fault disabled enabled						with Fault disabled enabled		
	Internal error pin	Error pin enable	Error pad	Internal error pin	Error pin enable	Error pad	Internal error pin	Error pin enable	Error pad
Disabled	NA OK	NA 1	NA OK	OK	1	OK	NA OK	NA 1	NA OK
Enabled	NA NOK	NA 1	NA NOK	OK	1	OK	NA OK	NA 1	NA OK

NOTE

The first time FCCU is in Configuration state, the EOUT pads are in a high-impedance state; thereafter, their states reflect the programmed EOUT protocol.

62.9.6.6 Dual-rail protocol

Dual-rail encoding is an alternate method for encoding bits. In contrast with classical encoding, where each signal carries a single-bit value, dual-rail encoded circuits use two wires to carry each bit. The encoding scheme is given in [Table 62-5](#) and the related timing diagram is given in [Figure 62-8](#) and [Figure 62-9](#).

Table 62-5. Dual-rail encoding

Logical state	Dual-rail encoding (output pins EOUT[1:0])	Note
non-faulty	10	toggling
non-faulty	01	
faulty	00	toggling
faulty	11	
reset	Hi impedance ¹	no toggling
configuration	same as NORMAL	toggling

1. Final value depends on the SoC setting at pad level.

As long as FCCU is in NORMAL or ALARM state, output will show "non-faulty" signal. Output pins EOUT[0:1] will toggle between 01 and 10 with a given frequency of 61 Hz.

In the RESET phase the output pins are set as "high impedance".

Note

[Figure 62-8](#) and [Figure 62-9](#) are formatted to display the behavior in all four phases (reset, normal, error, and config), not to imply transitions between one phase to another. In particular, transition from error phase to config phase is not possible.

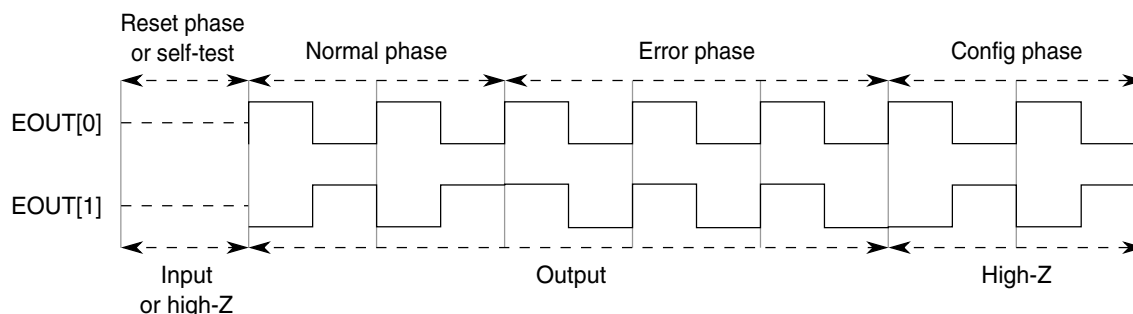


Figure 62-8. Dual-rail protocol (slow switching mode)

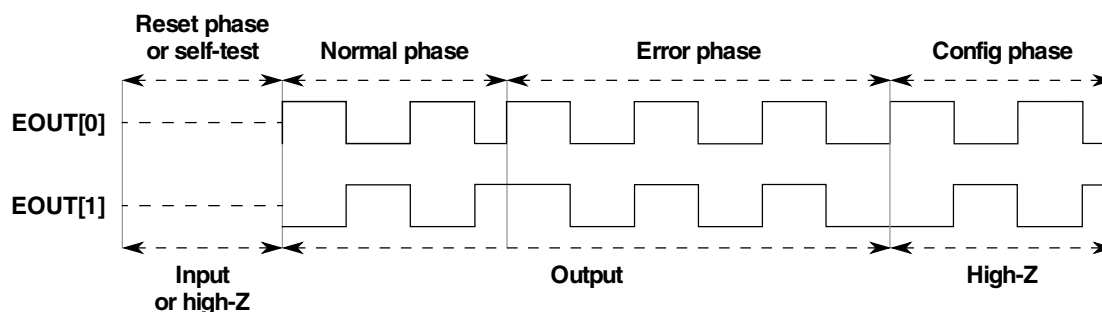


Figure 62-9. Dual-rail protocol (fast switching mode)

62.9.6.7 Time-switching protocol

The encoding scheme is given in [Table 62-6](#) and the related timing diagram is given in [Figure 62-10](#).

Table 62-6. Time-switching encoding

Logical state	Time-switching encoding (output pins EOUT[1:0])	Note
non-faulty	10	toggling
non-faulty	01	
faulty	10	no toggling
reset	high-Z ¹	no toggling
configuration	same as NORMAL	toggling

1. Final value depends on device specific settings at pad level.

As long as FCCU is in NORMAL or ALARM state, outputs will show "non-faulty" signal. Output pins #0, #1 will toggle between 01 and 10 with frequency of 61 Hz.

In the FAULT state, the output pin EOUT[0] is set as low.

In Time Switching mode the second output (EOUT[1]) is the inverted signal of first output (EOUT[0]).

In the RESET phase the output pins are set as "high impedance".²

Note

[Figure 62-10](#) is formatted to display the behavior in all four phases (reset, normal, error, and config), not to imply transitions between one phase to another. In particular, transition from error phase to config phase is not possible.

2. Actual value depends on device specific settings at pad level.

Functional description

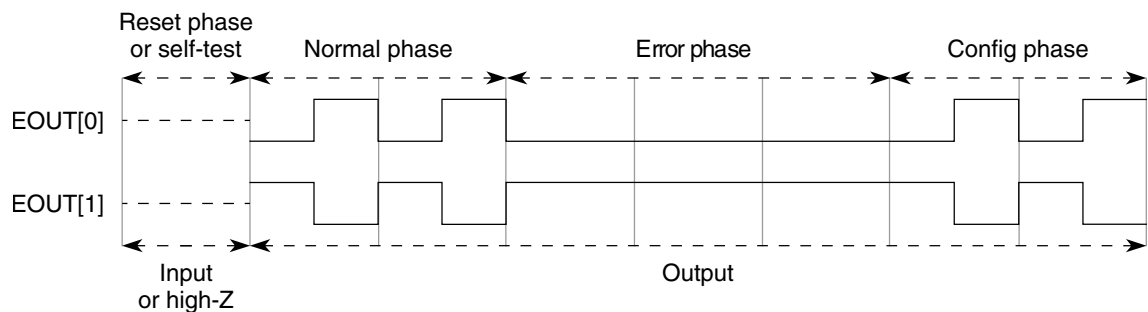


Figure 62-10. Time-switching protocol

62.9.6.8 Bi-stable protocol

The encoding scheme is given in [Table 62-7](#) and the related timing diagram is given in [Figure 62-11](#).

Table 62-7. Bi-stable encoding

Logical state	Bi-stable encoding (output pins EOUT[1:0])	Note
non-faulty	01	no toggling
faulty	10	no toggling
reset	high-Z ¹	no toggling
configuration	same as NORMAL	no toggling

1. Final value depends on device specific settings at pad level.

In the FAULT state, the faulty logical state is indicated. In NORMAL or ALARM state, "no-faulty" state is indicated. In Bi-stable mode the second output (EOUT[1]) is the inverted signal of first output (EOUT[0]).

In the RESET phase the output pins are set as "high impedance".³

Note

[Figure 62-11](#) is formatted to display the behavior in all four phases (reset, normal, error, and config), not to imply transitions between one phase to another. In particular, transition from error phase to config phase is not possible.

3. Actual value depends on device specific settings at pad level.

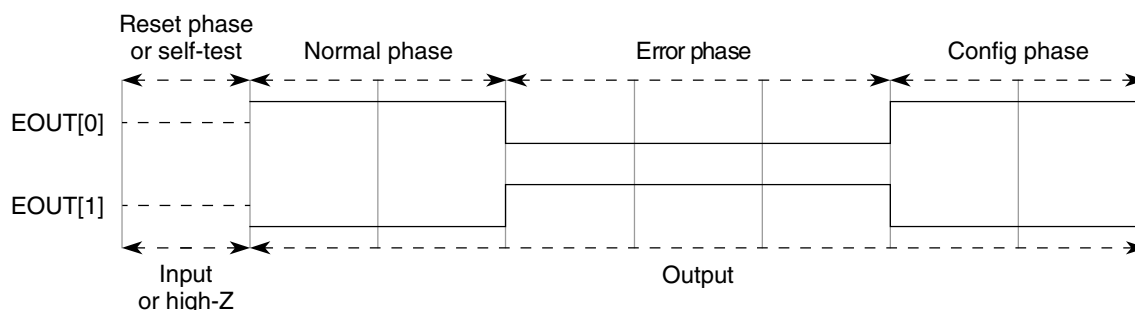


Figure 62-11. Bi-stable protocol

62.9.7 Error signal flow

See the chip-specific FCCU information for the diagram of the signal flow.

62.10 Register descriptions

The FCCU registers are listed in the table below. Any address offset not explicitly mentioned in this table is reserved.

The FCCU supports word (32-bit), half-word (16-bit), and byte (8-bit) read and write accesses.

Follow these register-access guidelines:

- Don't read from or write to any addresses that are not shown in the following table. Doing so may result in a transfer error.
- Don't write to any of the configuration registers unless FCCU is in Configuration state. Doing so results in a transfer error.
- Don't write to FCCU_TRANS_LOCK or FCCU_PERMNT_LOCK unless your code runs in Supervisor mode. Doing so results in a transfer error.

All the registers accessible in write mode only in CONFIG state are referred as configuration registers. These configuration registers return to the default value after configuration watchdog timer expires. These are the registers protected by FCCU_TRANS_LOCK and FCCU_PERMNT_LOCK registers. The configuration register setting has effect only when the FCCU state exits from the CONFIG state.

Register descriptions

For each possible NCF failure source, a different reaction—including no reaction—is configurable through the use of NMI, IRQ, and reset selection registers. It is not possible for a single event upset to switch off all reactions on failures as implementation is per fault source (but it will be possible to switch them all off by SW if intended). Failures themselves are not able to disable all reactions and indications.

The FCCU is not reset by functional resets.

FCCU memory map

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/page
0	Control (FCCU_CTRL)	32	R/W	0000_00C0h	62.10.1/3096
4	Control Key (FCCU_CTRLK)	32	W	0000_0000h	62.10.2/3098
8	Configuration (FCCU_CFG)	32	R/W	See section	62.10.3/3099
1C	Noncritical Fault Configuration (FCCU_NCF_CFG0)	32	R/W	See section	62.10.4/3102
20	Noncritical Fault Configuration (FCCU_NCF_CFG1)	32	R/W	See section	62.10.4/3102
4C	Noncritical Fault State Configuration (FCCU_NCFS_CFG0)	32	R/W	See section	62.10.5/3103
50	Noncritical Fault State Configuration (FCCU_NCFS_CFG1)	32	R/W	See section	62.10.5/3103
54	Noncritical Fault State Configuration (FCCU_NCFS_CFG2)	32	R/W	See section	62.10.5/3103
80	Noncritical Fault Status (FCCU_NCF_S0)	32	R/W	0000_0000h	62.10.6/3104
84	Noncritical Fault Status (FCCU_NCF_S1)	32	R/W	0000_0000h	62.10.6/3104
90	Noncritical Fault Key (FCCU_NCFK)	32	W	0000_0000h	62.10.7/3106
94	Noncritical Fault Enable (FCCU_NCF_E0)	32	R/W	See section	62.10.8/3107
98	Noncritical Fault Enable (FCCU_NCF_E1)	32	R/W	See section	62.10.8/3107
A4	Noncritical Fault Timeout Enable (FCCU_NCF_TOE0)	32	R/W	See section	62.10.9/3108
A8	Noncritical Fault Timeout Enable (FCCU_NCF_TOE1)	32	R/W	See section	62.10.9/3108
B4	Noncritical Fault Timeout (FCCU_NCF_TO)	32	R/W	See section	62.10.10/3109
B8	Configuration-State Timer Interval (FCCU_CFG_TO)	32	R/W	See section	62.10.11/3109
BC	IO Control (FCCU_EINOUT)	32	R/W	See section	62.10.12/3110

Table continues on the next page...

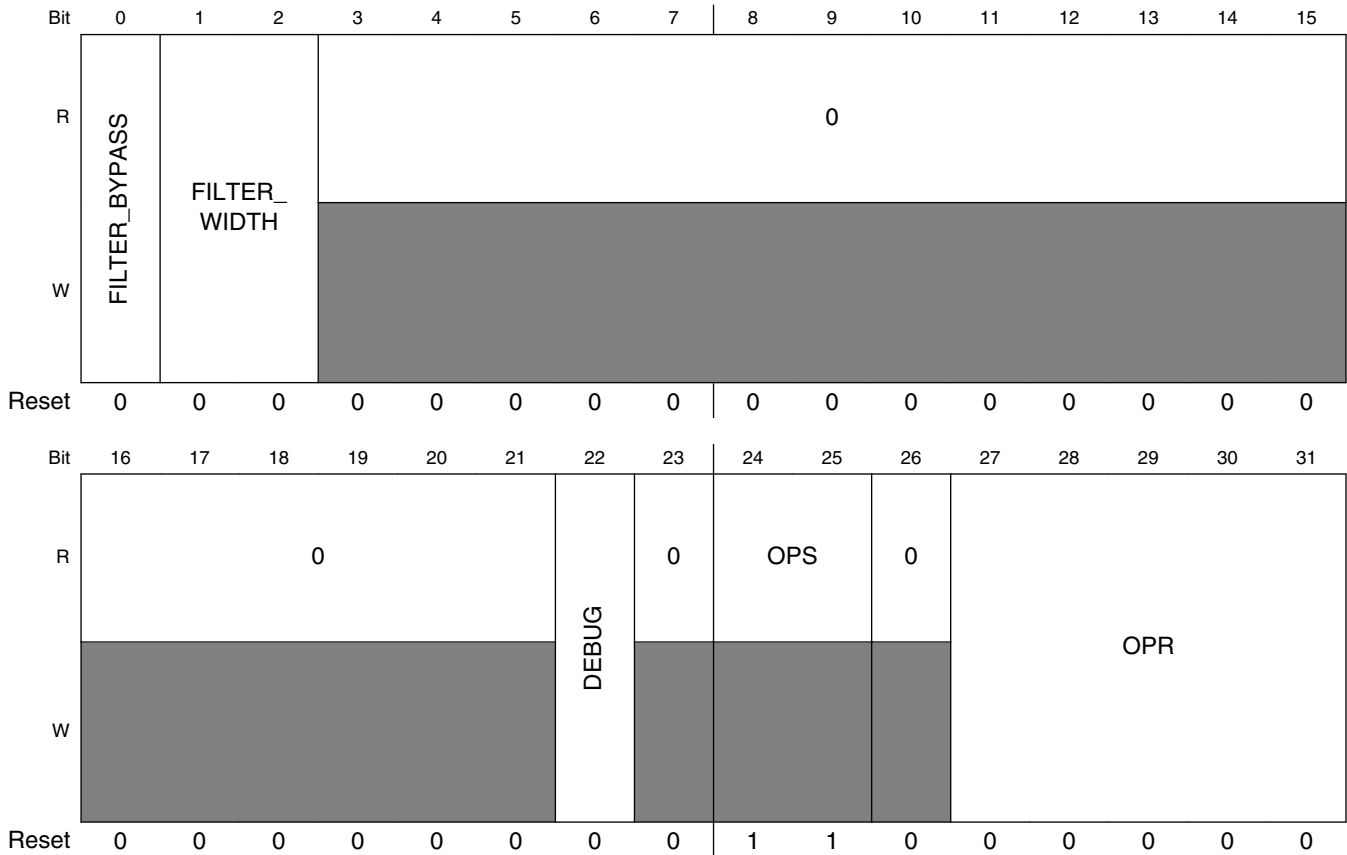
FCCU memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
C0	Status (FCCU_STAT)	32	R	0000_0010h	62.10.13/ 3112
C4	NA Freeze Status (FCCU_N2AF_STATUS)	32	R	0000_0000h	62.10.14/ 3114
C8	AF Freeze Status (FCCU_A2FF_STATUS)	32	R	0000_0000h	62.10.15/ 3115
CC	NF Freeze Status (FCCU_N2FF_STATUS)	32	R	0000_0000h	62.10.16/ 3116
D0	FA Freeze Status (FCCU_F2A_STATUS)	32	R	0000_0000h	62.10.17/ 3117
DC	Noncritical Fault Fake (FCCU_NCFF)	32	R/W	0000_0000h	62.10.18/ 3118
E0	IRQ Status (FCCU_IRQ_STAT)	32	R/W	0000_0000h	62.10.19/ 3119
E4	IRQ Enable (FCCU_IRQ_EN)	32	R/W	0000_0000h	62.10.20/ 3120
E8	XTMR (FCCU_XTMR)	32	R	0000_0000h	62.10.21/ 3121
F0	Transient Configuration Lock (FCCU_TRANS_LOCK)	32	W	0000_0000h	62.10.22/ 3122
F4	Permanent Configuration Lock (FCCU_PERMNT_LOCK)	32	W	0000_0000h	62.10.23/ 3123
F8	Delta T (FCCU_DELTA_T)	32	R/W	0000_0000h	62.10.24/ 3123
FC	IRQ Alarm Enable (FCCU_IRQ_ALARM_EN0)	32	R/W	0000_0000h	62.10.25/ 3124
100	IRQ Alarm Enable (FCCU_IRQ_ALARM_EN1)	32	R/W	0000_0000h	62.10.25/ 3124
10C	NMI Enable (FCCU_NMI_EN0)	32	R/W	0000_0000h	62.10.26/ 3125
110	NMI Enable (FCCU_NMI_EN1)	32	R/W	0000_0000h	62.10.26/ 3125
11C	Bistable Fault-Output (EOUT) Mode Signaling Enable (FCCU_EOUT_SIG_EN0)	32	R/W	0000_0000h	62.10.27/ 3126
120	Bistable Fault-Output (EOUT) Mode Signaling Enable (FCCU_EOUT_SIG_EN1)	32	R/W	0000_0000h	62.10.27/ 3126

62.10.1 Control (FCCU_CTRL)

Initiates and indicates the status of operations, indicates that FCCU successfully loaded configuration data from the NVMCFG signals, controls glitch-filter settings, and enables Debug mode.

Address: 0h base + 0h offset = 0h



FCCU_CTRL field descriptions

Field	Description
0 FILTER_ BYPASS	Filter bypass 0 glitch filter not bypassed 1 glitch filter bypassed
1–2 FILTER_WIDTH	Filter width 00 filters glitches up to 50 μs 01 filters glitches up to 75 μs 10 filters glitches up to 100 μs 11 filters glitches up to 100 μs

Table continues on the next page...

FCCU_CTRL field descriptions (continued)

Field	Description
3–21 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
22 DEBUG	Debug Mode Enable Specifies whether Debug mode is enabled. If so, FCCU enters Debug mode when the Debug signal is asserted. When FCCU enters Debug mode, it halts operation and remains in the state it was in before it entered Debug mode. NOTE: FOSU doesn't halt when FCCU enters Debug mode. Therefore, FOSU can still cause a reset if a fault occurs while FCCU is in Debug mode. 0 Disabled 1 Enabled
23 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
24–25 OPS	Operation status. This bit can be read and cleared (via OP15 operation) by the software. 00 Idle 01 In progress 10 Aborted 11 Successful
26 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
27–31 OPR	Operation Run Initiates operations that perform actions such as putting FCCU in Configuration state or reading a timer. For information on how to run operations, see Run operations. FCCU ignores any write to this field while the operation status (OPS) is In Progress. After completion of an operation, FCCU sets this field to OP0. The following events result in an operation status (OPS) of Aborted: - Writing to a Noncritical Fault Status (FCCU_NCF_Sn) register (which automatically initiates the OP12 operation) without first successfully unlocking the register (NCFK) - Initiating an OP1 operation when FCCU is not in Normal state or the configuration is locked - Initiating an OP1, OP2, or OP31 operation without first unlocking the operation (CTRLK) 00000 OP0—No operation 00001 OP1—Applies only when the configuration is unlocked, when FCCU is in Normal state, and immediately after you unlock the operation (CTRLK). Put FCCU in Configuration state. 00010 OP2—Applies only immediately after you unlock the operation (CTRLK). Put FCCU in Normal state. 00011 OP3—Capture FCCU's state and the states of the EOUT signals in the Status (FCCU_STAT) register. 00100 OP4—Read the FCCU frozen status flags (NA Freeze Status (FCCU_N2AF_STATUS)). 00101 OP5—Read the FCCU frozen status flags (AF Freeze Status (FCCU_A2FF_STATUS)). 00110 OP6—Read the FCCU frozen status flags (NF Freeze Status (FCCU_N2FF_STATUS)). 00111 OP7—Read the FCCU frozen status flags (FA Freeze Status (FCCU_F2A_STATUS)). 01000 Reserved 01001 Reserved

Table continues on the next page...

FCCU_CTRL field descriptions (continued)

Field	Description
	01010 OP10—Read the Noncritical Fault Status (FCCU_NCF_Sn) register.
	01011 Reserved
	01100 OP12—Do not initiate this operation; it is automatically initiated by FCCU. A Noncritical Fault Status (FCCU_NCF_Sn) register status clear operation is in progress.
	01101 OP13—Clear the freeze status registers (see the freeze registers).
	01110 OP14—Do not initiate this operation; it is automatically initiated by FCCU. A Configuration-state timeout is in progress. For more information, see Configuration registers .
	01111 OP15—Set the operation status (OPS) to Idle.
	10000 Reserved
	10001 OP17—Read the Alarm-state timer (XTMR).
	10010 Reserved
	10011 OP19—Read the Configuration-state timer (XTMR).
	10100 Reserved
	10101–11110 Forbidden. Writing any of these values returns an operation status (OPS) of Aborted with no side effect.
	11111 OP31—Applies only immediately after you unlock the operation (CTRLK). Load configuration data from the NVMCFG signals to certain fields in the Configuration (FCCU_CFG) register, but without resetting FCCU (for testing and debugging). See FCCU_CFG register bit value sources (N and C) by event . The operation status (OPS) for this operation is always Successful.

62.10.2 Control Key (FCCU_CTRLK)

Writable only with a 32-bit write. Unlocks locked operations so you can initiate them (OPR). For information on how to unlock locked operations before you initiate them, see [Run an operation](#).

NOTE

Reading from this register always returns the value 0000_0000h.

Address: 0h base + 4h offset = 4h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R																																
W	CTRLK																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

FCCU_CTRLK field descriptions

Field	Description
0–31 CTRLK	Locked-Operation Control Key

FCCU_CTRLK field descriptions (continued)

Field	Description
	Unlocks locked operations so you can initiate them (OPR). For information on how to unlock locked operations before you initiate them, see Run an operation. NOTE: You must initiate an operation in the instruction that immediately follows the instruction that unlocks it; otherwise, the operation is again locked. 9137_56AFh: Unlock OP1. 825A_132Bh: Unlock OP2. 29AF_8752h: Unlock OP31. Any other value: Do nothing.

62.10.3 Configuration (FCCU_CFG)

Writable only when FCCU is in Configuration state. Specifies the global configuration for FCCU.

The values of the bits in this register are affected by writes as well as any of the following events:

- The chip resets FCCU.
- FCCU automatically leaves Configuration state and enters Normal state because the configuration-timer interval ([TO](#)) expires. This is called a Configuration-state timeout.
- You run an OP31 operation.

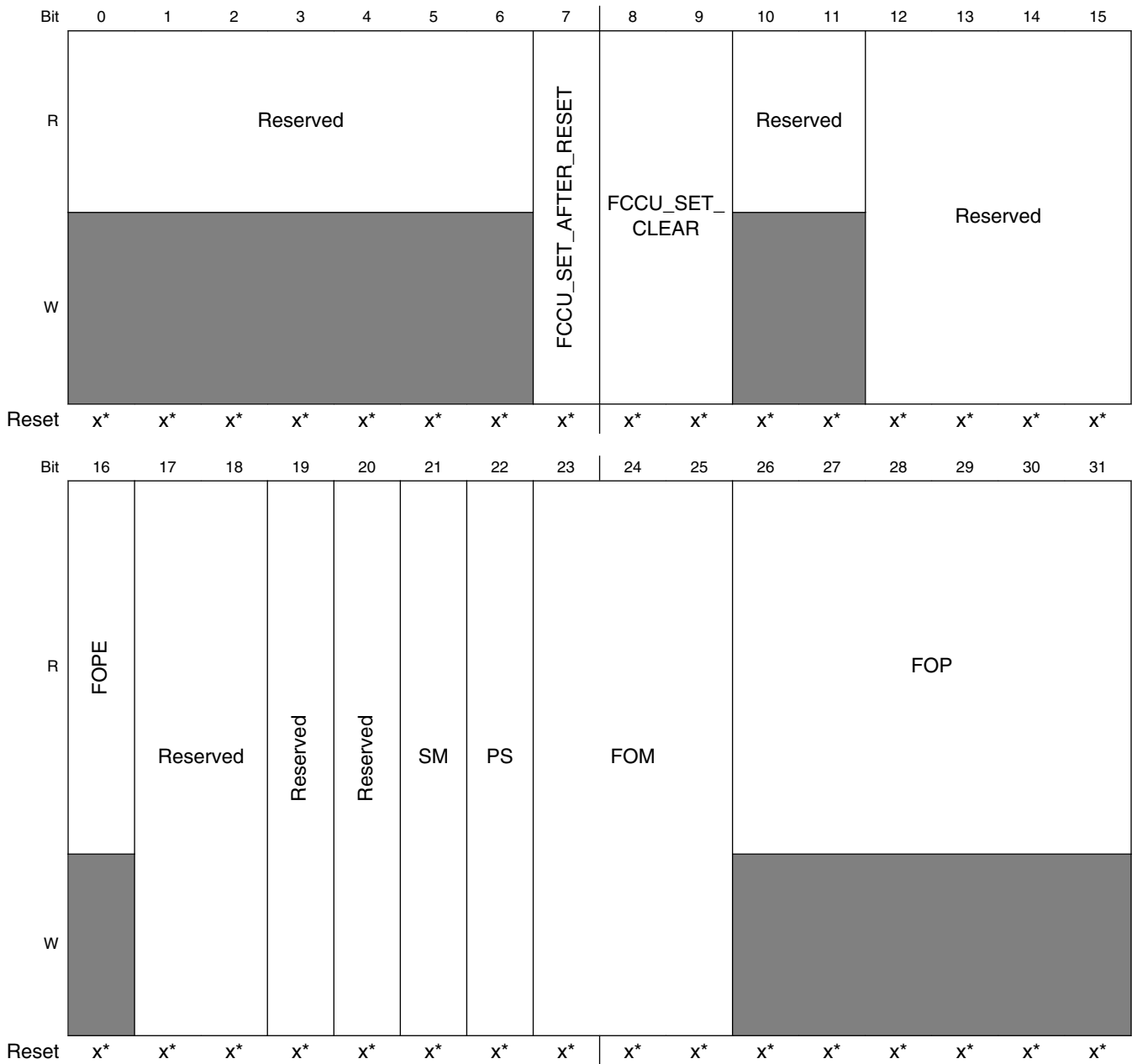
For information on the sources of bit values for these events, see [FCCU_CFG register bit value sources \(N and C\) by event](#).

NOTE

If you specify a new value for any of the fields in this register that affect the EOUT signals while the fault-output (EOUT) timer is running (FCCU is indicating a fault on the EOUT signals), FCCU doesn't use the new settings you specified until after the fault-output (EOUT) timer expires (FCCU stops indicating a fault on the EOUT signals).

Register descriptions

Address: 0h base + 8h offset = 8h



- * Notes:
- For information on this register's reset value, see [FCCU_CFG register bit value sources \(N and C\) by event](#). x = Undefined at reset.

FCCU_CFG field descriptions

Field	Description
0–6 Reserved	This field is reserved.

Table continues on the next page...

FCCU_CFG field descriptions (continued)

Field	Description
7 FCCU_SET_AFTER_RESET	Fault-Output (EOUT) Activate For fault-output (EOUT) signaling, controls whether the EOUT signals are active. 0 Inactive (the EOUT signals are in a high-impedance state) 1 Active (the EOUT signals indicate FCCU's condition)
8–9 FCCU_SET_CLEAR	Fault-Output (EOUT) Control Applies only when the EOUT signals are active (FCCU_SET_AFTER_RESET). Controls whether the fault-output (EOUT) signals are controlled by FCCU's finite state machine (FSM). 00 Controlled by the FSM 01 Always low 10 Controlled by the FSM 11 High until a fault occurs on a channel, regardless of whether that fault is disabled; thereafter, controlled by the FSM. Note: FCCU ignores an attempt to write this value if the fault-output (EOUT) timer is already running.
10–11 Reserved	This field is reserved.
12–15 Reserved	This field is reserved. Always write the reset value to this field.
16 FOPE	Fault-Output (EOUT) Prescaler Extension For fault-output (EOUT) signaling, specifies the most significant bit of the derived prescaler that controls the signaling frequency. For more information, see the description of the FOP field.
17–18 Reserved	This field is reserved. Always write the reset value to this field.
19 Reserved	This field is reserved. Always write the reset value to this field.
20 Reserved	This field is reserved. Always write the reset value to this field.
21 SM	Fault-Output (EOUT) Switching Mode For fault-output (EOUT) signaling, controls how quickly the signals switch between faulty and nonfaulty indication. Applies only to Dual-Rail and Time-Switching fault-output modes. 0 Slow: No EOUT frequency violation during the FCCU state transition (Normal to Fault or vice versa, and Configuration to Normal). The indication transition occurs after a maximum delay equal to the duration of the semiperiod of the EOUT frequency. 1 Fast: The indication transition (Normal to Fault or vice versa, and Configuration to Normal) occurs immediately. A pulse with the minimum duration corresponding to the CLKSAFE period can occur in this mode. It implies a frequency violation of the EOUT protocol.
22 PS	Fault-Output (EOUT) Polarity Selection For fault-output (EOUT) signaling, controls the polarity of the signals for fault-output-mode indications that hold the signals low or high (versus toggling them or placing them in a high-impedance state). Applies only to Time-Switching fault-output mode (for faulty and configuration indications) and Bistable fault-output mode (for all indications). 0 For the faulty indication, EOUT1 is high, and EOUT0 is low. 1 For the faulty indication, EOUT1 is low, and EOUT0 is high.

Table continues on the next page...

FCCU_CFG field descriptions (continued)

Field	Description
23–25 FOM	Fault-Output (EOUT) Mode For fault-output (EOUT) signaling, controls the protocol of the signaling. NOTE: In the test modes, a simple double-stage resynchronization stage is used to resynchronize the EOUT input/outputs with CLKSAFE. 000 Dual-Rail (default state) [EOUT[1:0]= outputs] 001 Time-Switching [EOUT[1:0]= output to be used] 010 Bistable 011 Reserved 100 Reserved 101 Test 0 (controlled by the FCCU_EINOUT register; EOUT1 is an output; EOUT0 is an input) 110 Test 1 (controlled by the FCCU_EINOUT register; EOUT1 and EOUT0 are both outputs) 111 Test 2 (controlled by the FCCU_EINOUT register; EOUT1 is an input; EOUT0 is an output)
26–31 FOP	Fault-Output (EOUT) Prescaler For fault-output (EOUT) signaling, specifies the least significant bits of the derived prescaler that controls the signaling frequency. The derived prescaler is the concatenation of FOPE and FOP and is represented by {FOPE, FOP}. The following equation defines the signaling frequency in terms of the CLKSAFE frequency ($\text{CLKSAFE}_{\text{freq}}$) and the derived prescaler: $\text{EOUT}_{\text{freq}} = \text{CLKSAFE}_{\text{freq}} / (((\text{FOPE, FOP}) + 1) \times 2 \times 2048)$ The following example values and meanings are for the derived (7-bit) prescaler, not FOP alone. The meanings show the derived prescaler followed by the corresponding value by which the input clock frequency is then divided. 0000000: 0; divide by 2×2048 (4096) 0000001: 1; divide by 4×2048 (8192) 0000010: 2; divide by 6×2048 (12,288) ... 0001000: 8; divide by 18×2048 (36,864) ... 1111101: 125; divide by 252×2048 (516,096) 1111110: 126; divide by 254×2048 (520,192) 1111111: 127; divide by 256×2048 (524,288) NOTE: Writing to FCCU_CFG sets FOP to 0.

62.10.4 Noncritical Fault Configuration (FCCU_NCF_CFGn)

Contains the configuration of noncritical faults in terms of fault-recovery management. The configuration depends on the type of signaling of a fault event. Hardware-recoverable faults should be configured only if a previous latching stage captures and holds the physical fault; otherwise, the fault can be lost. All other faults should be configured as software faults.

NOTE

These registers are writable only when FCCU is in Configuration state.

Noncritical Fault Configuration (FCCU_NCF_CFGn) register (value of n)	Offset	NCFCx fields (value of x)	
		Most significant (leftmost) bit	Least significant (rightmost) bit
0	1Ch	31	0
1	20h	63	32

Address: 0h base + 1Ch offset + (4d × i), where i=0d to 1d

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R																																
W	NCFCx																															
Reset	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*

* Notes:

- The reset value is chip-specific. See the chip-specific FCCU information.x = Undefined at reset.

FCCU_NCF_CFGn field descriptions

Field	Description
0–31 NCFCx	Noncritical Fault Configuration. Defines the fault recovery mode. Hardware-recoverable faults are self-recovered (status flag clearing and related) if the root cause (input fault) has been removed. Software-recoverable faults are recovered (status flag clearing) by software clearing of the related status flag. For the mapping of the NCFCx fields among registers, see the earlier table in this register description. 0 Hardware-recoverable fault 1 Software-recoverable fault

62.10.5 Noncritical Fault State Configuration (FCCU_NCFS_CFGn)

The FCCU_NCFS_CFGx registers contain the configuration of each noncritical fault in terms of fault reaction (functional reset request pulse) when it is the root cause for the FAULT state transition.

NOTE

These registers are writable only when the FCCU is in the CONFIG state.

Table 62-16 shows FCCU configuration register channels.

Table 62-16. FCCU NCFS configuration register channels

Address offset	Register name	Channel range (x)
4Ch	FCCU_NCFS_CFG0	NCFSCx[15:0]
50h	FCCU_NCFS_CFG1	NCFSCx[31:16]
54h	FCCU_NCFS_CFG2	NCFSCx[47:32]

Address: 0h base + 4Ch offset + (4d × i), where i=0d to 2d

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R																																
W	NCFSCx																															
Reset	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	

* Notes:

- The reset value is chip-specific. See the chip-specific FCCU information.x = Undefined at reset.

FCCU_NCFS_CFGn field descriptions

Field	Description
0–31 NCFSCx	Noncritical fault state configuration See Table 62-16 for register offset to channel number relationship. 00 No reset reaction 01 Reset request pulse (FAULT state reaction) 10 Reset request pulse (FAULT state reaction) 11 No reset reaction

62.10.6 Noncritical Fault Status (FCCU_NCF_Sn)

The FCCU_NCF_Sx register contains the latched fault indication collected from the noncritical fault sources. Faults are latched also in the CONFIG state and independently from the enabling or reactions programmed for the NCF.

No reactions are executed until the FCCU moves in the NORMAL state.

FCCU reacts and moves from the NORMAL state into the ALARM state only if the respective enable bit for a fault is set in the FCCU_NCF_Ex register and the respective enable bit for the timeout is set in the FCCU_TOEx register.

FCCU reacts and moves from the NORMAL or ALARM state into the FAULT state if the respective enable bit for a fault is set in the FCCU_NCF_Ex register and the respective enable bit for the timeout is disabled in the FCCU_TOEx register.

FCCU reacts and moves from the ALARM state into the FAULT state if the timeout (FCCU_TO register) is elapsed before recovering from the fault.

The timeout is stopped only when the FCCU returns in the NORMAL state.

The FCCU_NCF_Sx register is encoded respectively into the N2FF_STATUS or A2FF_STATUS register to freeze the entry condition in the FAULT state. The status bits of the FCCU_NCF_Sx register, configured as SW recoverable faults, can be cleared by the following locked sequence:

1. Write the proper key into the FCCU_NCFK register.
2. Clear the status (flag) bit NCFSx => the opcode OP12 is automatically set into the FCCU_CTRL.OPR field.
3. Wait for the completion of the operation (FCCU_CTRL.OPS field).
4. Read the FCCU_NCF_Sx register in order to verify the effective deletion and in case of failure to repeat the sequence.

As result of the above sequence, in addition the FAULT interface provides support to clear the external fake FAULT root .

NOTE

There should not be any other operation in between the above steps.

The FCCU moves from the FAULT or ALARM state into the NORMAL state if all the source faults that caused the transition into the FAULT state have been removed (HW recoverable fault) or cleared via SW (SW recoverable fault). In case of nested faults that are not all recovered, the FCCU will remain in the FAULT or ALARM state.

The SW application executes the FCCU_NCF_Sx read operation by the following sequence:

1. Set the OP10 operation into the FCCU_CTRL.OPR field.
2. Wait for the completion of the operation (FCCU_CTRL.OPS field).
3. Read the FCCU_NCF_Sx register.

In case of re-configuration of the FCCU (CONFIG state), before to return in NORMAL state the pending status bits into the FCCU_NCF_Sx must be cleared in order to avoid a false transition in ALARM/FAULT state.

The following faults are ignored:

- to write a wrong key into the FCCU_NCFK register
- to attempt to clear a HW recoverable fault

Noncritical Fault Status (FCCU_NCF_Sn) register (value of n)	Offset	NCFSx fields (value of x)	
		Most significant (leftmost) bit	Least significant (rightmost) bit
0	80h	31	0
1	84h	63	32

Register descriptions

Address: 0h base + 80h offset + (4d × i), where i=0d to 1d

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	NCFSx																															
W	w1c																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

FCCU_NCF_Sn field descriptions

Field	Description
0–31 NCFSx	Noncritical fault status. The status bits related to the noncritical fault configured as HW recoverable faults are read-only and the flag is self cleared when the fault source is removed. For the mapping of the NCFSx fields among registers, see the earlier table in this register description. 0 No "noncritical" fault latched 1 "Noncritical" fault latched

62.10.7 Noncritical Fault Key (FCCU_NCFK)

The FCCU_NCFK register implements the key access to clear the status flags of the FCCU_NCF_Sx register.

The status bits of the FCCU_NCF_Sx register, configured as SW recoverable faults, can be cleared by the following locked sequence:

1. Write the key into the FCCU_NCFK register.
2. Clear the status (flag) bit NCFSx .

The key must be written for each FCCU_NCF_Sx clear operation.

The FCCU_NCFK register is not readable; a 0000_0000h value is always returned in case of read operation.

Address: 0h base + 90h offset = 90h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0																															
W	NCFK																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

FCCU_NCFK field descriptions

Field	Description
0–31 NCFK	Noncritical fault key = AB34_98FEh

62.10.8 Noncritical Fault Enable (FCCU_NCF_En)

The FCCU_NCF_En registers enable the fault sources to allow a transition from the NORMAL into the FAULT or ALARM state. In case of fault masking, the respective status bit into the FCCU_NCF_Sn register is set (for debugging purposes), only the reaction is masked.

NOTE

These registers are writable only when the FCCU is in the CONFIG state.

NOTE

Any enabled fault should be programmed to result in a defined action. For example, set up an ALARM IRQ action by programming the IRQ Alarm Enable Register (FCCU_IRQ_ALARM_ENn).

Noncritical Fault Enable (FCCU_NCF_En) register (value of n)	Offset	NCFEx fields (value of x)	
		Most significant (leftmost) bit	Least significant (rightmost) bit
0	94h	31	0
1	98h	63	32

Address: 0h base + 94h offset + (4d × i), where i=0d to 1d

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R																																
W																																
	NCFEx																															
Reset	X*	X*	X*	X*	X*	X*	X*	X*	X*	X*	X*	X*	X*	X*	X*	X*	X*	X*	X*	X*	X*	X*	X*	X*	X*	X*	X*	X*	X*	X*	X*	

* Notes:

- The reset value is chip-specific. See the chip-specific FCCU information.x = Undefined at reset.

FCCU_NCF_En field descriptions

Field	Description
0–31 NCFEx	Noncritical fault enable For the mapping of the NCFEx fields among registers, see the earlier table in this register description. 0 No actions following the respective noncritical fault assertion 1 FCCU moves to ALARM or FAULT state

62.10.9 Noncritical Fault Timeout Enable (FCCU_NCF_TOEn)

The FCCU_NCF_TOEx registers enable a transition from the NORMAL state into the ALARM state if the respective noncritical fault is enabled (NCFEx and NCFTOEx are set). In case the respective timeout is disabled (NCFTOEx is cleared) and the noncritical fault is enabled (NCFEx is set) the FCCU moves into the FAULT state if the related noncritical fault is asserted. The timer (preset with the timeout value defined by FCCU_TO register) is started when the FCCU moves into the ALARM state. If the fault is not recovered within the timeout the FCCU moves from the ALARM state to the FAULT state.

NOTE

These registers are writable only when the FCCU is in the CONFIG state.

Noncritical Fault Timeout Enable (FCCU_NCF_TOEn) register (value of n)	Offset	NCFTOEx fields (value of x)	
		Most significant (leftmost) bit	Least significant (rightmost) bit
0	A4h	31	0
1	A8h	63	32

Address: 0h base + A4h offset + (4d × i), where i=0d to 1d

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	NCFTOEx																															
W																																
Reset	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*

* Notes:

- The reset value is chip-specific. See the chip-specific FCCU information.x = Undefined at reset.

FCCU_NCF_TOEn field descriptions

Field	Description
0–31 NCFTOEx	Fault timeout enable For the mapping of the NCFTOEx fields among registers, see the earlier table in this register description. 0 FCCU moves into the FAULT state if the respective fault is enabled 1 FCCU moves into the ALARM state if the respective fault is enabled

62.10.10 Noncritical Fault Timeout (FCCU_NCF_TO)

Defines the preset value of the timer for the recovery of enabled noncritical faults. Once FCCU enters Alarm state, following the assertion of a noncritical fault that is enabled (NCFEa and NCFTOEa are set), the timer starts the countdown.

If the fault is not recovered within the timeout period, the FCCU moves from the Alarm state to the Fault state. The alarm timeout value should be programmed to be less than FOSU_COUNT, or resets may be generated by FOSU timeouts.

NOTE

This register is writable only when FCCU is in Configuration state.

Address: 0h base + B4h offset = B4h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R																																
W	TO																															
Reset	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	

* Notes:

- The reset value is chip-specific. See the chip-specific FCCU information.x = Undefined at reset.

FCCU_NCF_TO field descriptions

Field	Description
0–31 TO	Noncritical Fault Timeout Timeout = (TO) × T _{RC16MHz}

62.10.11 Configuration-State Timer Interval (FCCU_CFG_TO)

Writable only when FCCU is in Normal, Alarm, or Fault state (not in Configuration state). Not accessible while a Configuration-state timeout (OP14 operation) is in progress. Controls the maximum amount of time that FCCU can be in Configuration state.

NOTE

When a Configuration-state timeout occurs, FCCU changes the value of this register to its reset value.

Register descriptions

Address: 0h base + B8h offset = B8h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0																															
W																TO																
Reset	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*

* Notes:

- The reset value is chip-specific. See the chip-specific FCCU information.x = Undefined at reset.

FCCU_CFG_TO field descriptions

Field	Description
0–28 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
29–31 TO	Configuration-State Timer Interval Controls the maximum amount of time (T_max) that FCCU can be in Configuration state according to this equation: $T_{\text{max}} = T_{16\text{MHz}} \times 2^{(\text{TO} + 10)}$ If you put FCCU in Configuration state and this timer interval expires (called a Configuration-state timeout), then FCCU: - Automatically leaves Configuration state and enters Normal state. - Changes the value of the Configuration (FCCU_CFG) register to its Configuration-state-timeout value and the value of each of the other configuration registers to its reset value. For information on the Configuration (FCCU_CFG) register's Configuration-state-timeout value, see FCCU_CFG register bit value sources (N and C) by event. For a list of configuration registers, see Configuration registers. NOTE: Make sure the Configuration-state timer interval is less than the FOSU module's timer interval; otherwise, a fault that occurs while FCCU is in Configuration state can cause FOSU to generate a chip reset. The FOSU timer interval (FOSU_COUNT) is chip-specific. See the chip-specific FCCU information. 000: 64 μs ... 111: 8192 μs

62.10.12 IO Control (FCCU_EINOUT)

The FCCU_EINOUT register allows the following operations typically in NORMAL state:

- to control the EOUT[1] output level when the FCCU is configured in "Test1" or "Test0" fault output mode (FCCU_CFG.FOM)
- to control the EOUT[0] output level when the FCCU is configured in "Test1" or "Test2" fault output mode (FCCU_CFG.FOM)
- to observe the EOUT[1:0] signals in input mode

[Table 62-36](#) shows Bi-stable encoding.

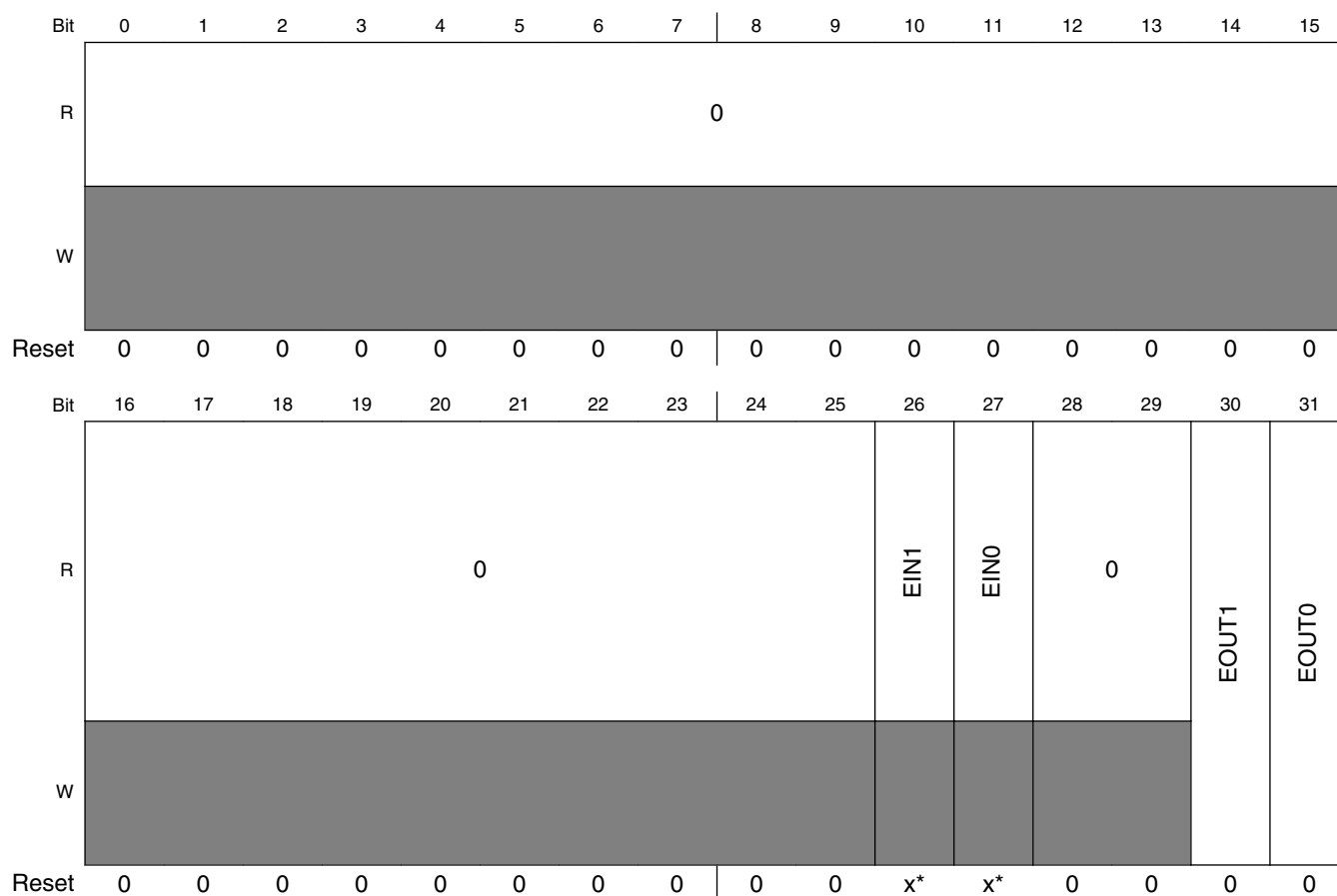
Table 62-36. Bi-stable encoding

Mode = FCCU_CFG.FOM	EOUT[0]	EOUT[1]
Test1	output	output
Test2	output	input
Test0	input	output

NOTE

Due to the resynchronization stage of the EOUT interface, there is a latency of a few CLKSAFE cycles following a write/read operation of the FCCU_EINOUT register.

Address: 0h base + BCh offset = BCh



* Notes:

- x = Undefined at reset.

FCCU_EINOUT field descriptions

Field	Description
0–25 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

Table continues on the next page...

FCCU_EINOUT field descriptions (continued)

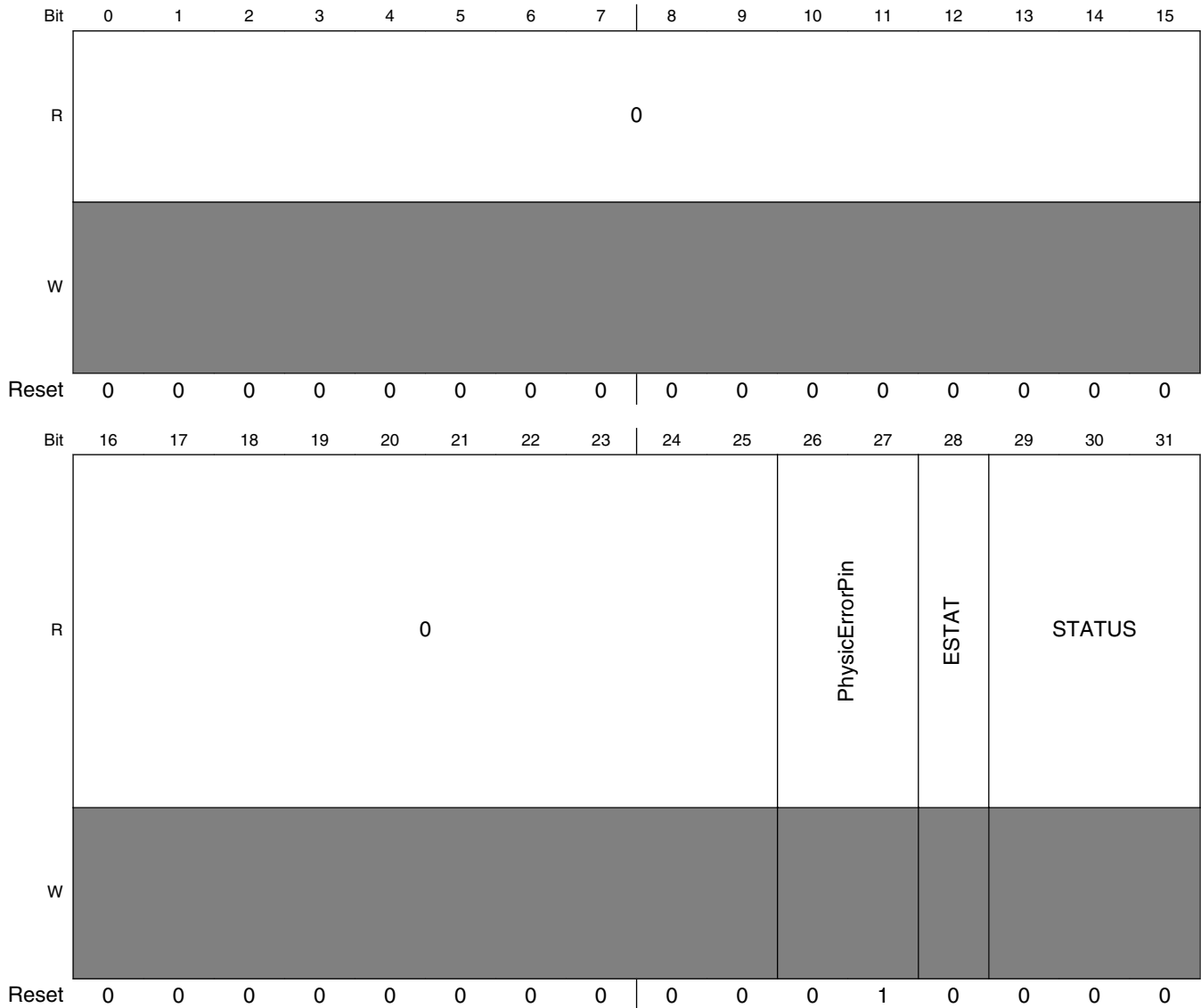
Field	Description
26 EIN1	Error Input 1 Applies only when the EOUT signals are active (FCCU_SET_AFTER_RESET). Indicates the state of the EIN1 signal. 0 Low 1 High
27 EIN0	Error Input 0 Applies only when the EOUT signals are active (FCCU_SET_AFTER_RESET). Indicates the state of the EIN0 signal. 0 Low 1 High
28–29 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
30 EOUT1	Error out 1 (significant only if the FCCU_CFG.FOM = Test1 or Test0 => EOUT[1] configured in output mode). The EOUT1 set/clear the respective EOUT[1] output signal if FCCU_CFG.FOM = 110 or 101, otherwise it is a "don't-care" value. NOTE: When the configuration watchdog timer expires, FCCU changes the value of this field to its reset value. 0 force EOUT[1] = 0 1 force EOUT[1] = 1
31 EOUT0	Error out 0 (significant only if the FCCU_CFG.FOM = Test1 or Test2 => EOUT[0] configured in output mode). The EOUT0 set/clear the respective EOUT[0] output signal if FCCU_CFG.FOM = 110 or 111, otherwise it is a "don't care" value. NOTE: When the configuration watchdog timer expires, FCCU changes the value of this field to its reset value. 0 force EOUT[0] = 0 1 force EOUT[0] = 1

62.10.13 Status (FCCU_STAT)

The FCCU_STAT register includes the FCCU status for debugging/test purposes. The FCCU finite state machine operates by the RC oscillator clock asynchronous with the system clock. The FCCU status read operation requires a safe mechanism operated by a HW/SW synchronization sequence. The SW application executes a FCCU status read operation by the following sequence:

1. Set the OP3 operation into the FCCU_CTRL.OPR field.
2. Wait for the completion of the operation (FCCU_CTRL.OPS field).
3. Read the FCCU status (FCCU_STAT register).

Address: 0h base + C0h offset = C0h

**FCCU_STAT field descriptions**

Field	Description
0–25 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
26–27 PhysicErrorPin	Fault Output (EOUT) States Applies only after successfully running an OP3 operation (OPR) and only when the EOUT signals are active (FCCU_SET_AFTER_RESET). Indicates the states that FCCU is driving on the EOUT signals. 00 EOUT1 is low; EOUT0 is low. 01 EOUT1 is low; EOUT0 is high. 10 EOUT1 is high; EOUT0 is low. 11 EOUT1 is high; EOUT0 is high.
28 ESTAT	Fault State

Table continues on the next page...

FCCU_STAT field descriptions (continued)

Field	Description
	Applies only after successfully running an OP3 operation (OPR). Indicates whether FCCU is in Fault state.
	0 Not in Fault state (in Normal, Alarm, or Configuration state)
	1 In Fault state
29–31 STATUS	FCCU State Applies only after successfully running an OP3 operation (OPR). Indicates FCCU's state.
	000 Normal
	001 Configuration
	010 Alarm
	011 Fault
	100 Reserved
	101 Reserved
	110 Reserved
	111 Reserved

62.10.14 NA Freeze Status (FCCU_N2AF_STATUS)

The N2AF_STATUS register contains a unique code to identify the "noncritical fault" source (fccu_ncf[x]) that caused the state transition from the NORMAL state to the ALARM state. In case of multiple fault conditions the fault source cannot be identified. This register is for test/debug purposes.

The SW application executes the N2AF_STATUS read operation by the following sequence:

1. Set the OP4 operation into the FCCU_CTRL.OPR field.
2. Wait for the completion of the operation (FCCU_CTRL.OPS field).
3. Read the N2AF_STATUS register.

The SW application executes the N2AF_STATUS clear operation by the following sequence:

1. Set the OP13 operation into the FCCU_CTRL.OPR field.
2. Wait for the completion of the operation (FCCU_CTRL.OPS field). (All the freeze registers are cleared by this operation.)

Address: 0h base + C4h offset = C4h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0																							NAFS								
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	

FCCU_N2AF_STATUS field descriptions

Field	Description
0–23 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
24–31 NAFS	Normal to Alarm Frozen Status 00000000 No transition from NORMAL to ALARM state 00000001 NORMAL to ALARM state transition cause => fccu_ncf[0] fault 00000010 NORMAL to ALARM state transition cause => fccu_ncf[1] fault 00000011 NORMAL to ALARM state transition cause => fccu_ncf[2] fault ... 10000000 NORMAL to ALARM state transition cause => fccu_ncf[127]/ fault 11111111 NORMAL to ALARM state transition cause => multiple fccu_ncf[]/ faults

62.10.15 AF Freeze Status (FCCU_A2FF_STATUS)

The A2FF_STATUS register contains a unique code to identify the timeout trigger (noncritical fault) that caused the state transition from the ALARM state to the FAULT state. In case of multiple fault conditions the fault source cannot be identified. This register is for test/debug purposes.

The SW application executes the A2FF_STATUS read operation by the following sequence:

1. Set the OP5 operation into the FCCU_CTRL.OPR field.
2. Wait for the completion of the operation (FCCU_CTRL.OPS field).
3. Read the A2FF_STATUS register.

The SW application executes the FCCU_AFFS clear operation by the following sequence:

1. Set the OP13 operation into the FCCU_CTRL.OPR field.
2. Wait for the completion of the operation (FCCU_CTRL.OPS field).

All the freeze registers are cleared by this operation.

Address: 0h base + C8h offset = C8h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0																					AF_SRC		AFFS								
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

FCCU_A2FF_STATUS field descriptions

Field	Description
0–21 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
22–23 AF_SRC	Fault source 00 No fault 01 Reserved 10 "Noncritical" fault 11 Multiple and/or "noncritical" faults
24–31 AFFS	Alarm to Fault Frozen Status 00h: No transition from ALARM to FAULT state 01h: ALARM to FAULT state transition cause => fccu_ncf[0]/ fault timeout fault 02h: ALARM to FAULT state transition cause => fccu_ncf[1]/ fault timeout fault 03h: ALARM to FAULT state transition cause => fccu_ncf[2]/ fault timeout fault ... 80h: ALARM to FAULT state transition cause => fccu_ncf[127]/ fault timeout fault FFh: ALARM to FAULT state transition cause => multiple fccu_ncf[]/ faults timeout faults

62.10.16 NF Freeze Status (FCCU_N2FF_STATUS)

The N2FF_STATUS register contains a unique code to identify the source of the noncritical fault that caused the state transition from the NORMAL state to the FAULT state. In case of multiple fault conditions the fault source cannot be identified. This register is for test/debug purposes. The SW application executes the N2FF_STATUS read operation by the following sequence:

1. Set the OP6 operation into the FCCU_CTRL.OPR field.
2. Wait for the completion of the operation (FCCU_CTRL.OPS field).
3. Read the N2FF_STATUS register.

The SW application executes the N2FF_STATUS clear operation by the following sequence:

1. Set the OP13 operation into the FCCU_CTRL.OPR field.
2. Wait for the completion of the operation (FCCU_CTRL.OPS field). (All the freeze registers are cleared by this operation.)

Address: 0h base + CCh offset = CCh

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0																					NF_SRC		NFFS								
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

FCCU_N2FF_STATUS field descriptions

Field	Description
0–21 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
22–23 NF_SRC	NF_SRC: Fault source 00 No fault 01 Reserved 10 "Noncritical" fault 11 Multiple "noncritical" faults
24–31 NFFS	Normal to Fault Frozen Status 00h: No transition from NORMAL to FAULT state 01h: NORMAL to FAULT state transition cause => fccu_ncf[0] fault 02h: NORMAL to FAULT state transition cause => fccu_ncf[1] fault 03h: NORMAL to FAULT state transition cause => fccu_ncf[2] fault ... 80h: NORMAL to FAULT state transition cause => fccu_ncf[127] fault FFh: NORMAL to FAULT state transition cause => multiple fccu_ncf[] faults

62.10.17 FA Freeze Status (FCCU_F2A_STATUS)

The F2AF_STATUS register contains a unique code to identify the source of the noncritical fault (fccu_ncf[x]) that caused the state transition from the FAULT state to the ALARM state. In case of multiple fault conditions the fault source cannot be identified. This register is for test/debug purposes.

The SW application executes the F2AF_STATUS read operation by the following sequence:

1. Set the OP7 operation into the FCCU_CTRL.OPR field.
2. Wait for the completion of the operation (FCCU_CTRL.OPS field).
3. Read the F2AF_STATUS register.

The SW application executes the F2AF_STATUS clear operation by the following sequence:

1. Set the OP13 operation into the FCCU_CTRL.OPR field.

Register descriptions

- Wait for the completion of the operation (FCCU_CTRL.OPS field). (All the freeze registers are cleared by this operation.)

Address: 0h base + D0h offset = D0h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0																						FAFS									
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	

FCCU_F2A_STATUS field descriptions

Field	Description
0–22 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
23–31 FAFS	Fault to Normal Frozen Status 00h: No transition from FAULT to ALARM state 01h: FAULT to ALARM state transition cause => fccu_ncf[0]/ fault 02h: FAULT to ALARM state transition cause => fccu_ncf[1]/ fault 03h: FAULT to ALARM state transition cause => fccu_ncf[2]/ fault ... 80h: FAULT to ALARM state transition cause => fccu_ncf[127]/ fault FFh: FAULT to ALARM state transition cause => multiple fccu_ncf[] faults

62.10.18 Noncritical Fault Fake (FCCU_NCFF)

The FCCU_NCFF register contains a unique code to set a noncritical fault in mutually exclusive mode by the external FAULT interface (signal setting). It allows the SW emulation of the noncritical faults, by the injection of the fault directly in the FAULT root, in order to verify the entire path and reaction. The reaction following a fake noncritical fault cannot be masked. The FCCU_NCFF is a write-only register with a set of codes corresponding to each noncritical fault injection.

Address: 0h base + DCh offset = DCh

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0																															
W																									FNCFC							
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

FCCU_NCFF field descriptions

Field	Description
0–24 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
25–31 FNCFC	FNCFC/ : Fake noncritical fault code NOTE: Only writing to this register, fake fault injection occurs, writing 00 and default value being zero give different results. 00h: Fake noncriticalfault injection at noncriticalfault source 0 01h: Fake noncriticalfault injection at noncriticalfault source 1 02h: Fake noncriticalfault injection at noncriticalfault source 2 ... 7Fh: Fake noncriticalfault injection at noncriticalfault source 127

62.10.19 IRQ Status (FCCU_IRQ_STAT)

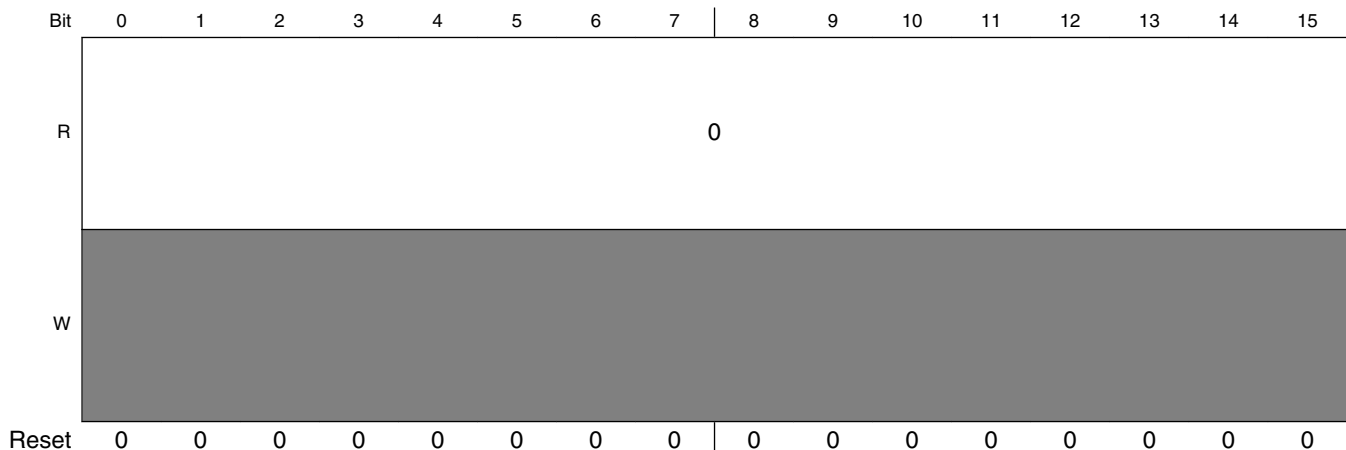
The FCCU_IRQ_STAT register provides the FCCU interrupt status related to the following events:

- Configuration timeout error
- Alarm interrupt
- NMI interrupt

The configuration timeout interrupt is asserted if the CFG_TO_STAT bit of the FCCU_IRQ_STAT register is set and the CFG_TO_IEN bit of the FCCU_IRQ_EN register is also set. It is cleared when a 1 is written to the CFG_TO_STAT bit.

The NMI and ALARM interrupts are asserted and cleared according to the FCCU state. The status bits of the FCCU_IRQ_STAT trace the status of the related interrupt lines.

Address: 0h base + E0h offset = E0h



Register descriptions

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0											Reserved	Reserved	NMI_STAT	ALRM_STAT	CFG_TO_STAT
W																w1c
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

FCCU_IRQ_STAT field descriptions

Field	Description
0–26 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
27 Reserved	This field is reserved.
28 Reserved	This field is reserved.
29 NMI_STAT	NMI Interrupt Status 0 NMI interrupt is OFF 1 NMI interrupt is ON
30 ALRM_STAT	Alarm Interrupt Status 0 Alarm interrupt is OFF 1 Alarm interrupt is ON
31 CFG_TO_STAT	Configuration Timeout Status 0 No configuration timeout error 1 Configuration timeout error

62.10.20 IRQ Enable (FCCU_IRQ_EN)

The FCCU_IRQ_EN register defines the FCCU interrupt enable register related to the following event:

- Configuration timeout error

The configuration timeout interrupt is asserted if the CFG_TO_STAT bit of the FCCU_IRQ_STAT register is set and the CFG_TO_IEN bit of the FCCU_IRQ_EN register is also set.

Address: 0h base + E4h offset = E4h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0												0	0	CFG_TO_IEN	
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

FCCU_IRQ_EN field descriptions

Field	Description
0–27 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
28 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
29–30 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
31 CFG_TO_IEN	Configuration Timeout Interrupt Enable 0 Configuration timeout interrupt disabled 1 Configuration timeout interrupt enabled

62.10.21 XTMR (FCCU_XTMR)

The FCCU_XTMR register contains the read values of the Alarm or Watchdog Timer. These timers are clocked by CLKSAFE.

The SW application executes the timer read operation by the following sequence:

1. Set any of the following operations into the FCCU_CTRL.OPR field:
 - OP17
 - OP19
2. Wait for the completion of the operation (FCCU_CTRL.OPS field).
3. Read the FCCU_XTMR register.

Table 62-46 shows Timer state/value.

Table 62-46. Timer state/value

TIMER	CONFIG state	NORMAL state	ALARM state	FAULT state
ALARM	00000000h	Initial value	Running	Idle/End of count
CFG	Running	0001FFFFh	0001FFFFh	0001FFFFh

Address: 0h base + E8h offset = E8h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	XTMR																															
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	

FCCU_XTMR field descriptions

Field	Description
0–31 XTMR	Alarm/Watchdog The current timer value is measured in CLKSAFE cycles. These bits can be read by the software.

62.10.22 Transient Configuration Lock (FCCU_TRANS_LOCK)

Writable only by code running in Supervisor mode. Temporarily locks and unlocks the configuration. Locking the configuration prevents FCCU from entering Configuration state.

Address: 0h base + F0h offset = F0h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	Reserved																															
W																							TRANSKEY									
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	

FCCU_TRANS_LOCK field descriptions

Field	Description
0–22 Reserved	This field is reserved.
23–31 TRANSKEY	Transient Configuration Lock Writable only by code running in Supervisor mode. Temporarily locks and unlocks the configuration. Locking the configuration prevents FCCU from entering Configuration state. BCh: Unlock. Any other value: Lock.

62.10.23 Permanent Configuration Lock (FCCU_PERMNT_LOCK)

Writable only by code running in Supervisor mode. Permanently locks the configuration, which prevents FCCU from entering Configuration state until FCCU is reset.

Address: 0h base + F4h offset = F4h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	Reserved																								PERMNTKEY							
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

FCCU_PERMNT_LOCK field descriptions

Field	Description
0–22 Reserved	This field is reserved.
23–31 PERMNTKEY	Permanent Configuration Lock Writable only by code running in Supervisor mode. Permanently locks the configuration, which prevents FCCU from entering Configuration state until FCCU is reset. FFh: Lock. Any other value: Do nothing.

62.10.24 Delta T (FCCU_DELTA_T)

The FCCU_DELTA_T register is used for programming the value of delta_T constant, in microseconds.

NOTE

This register can be written only when the FCCU is in CONFIG state.

NOTE

Reserved bits should always be written as all 0's.

Address: 0h base + F8h offset = F8h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	Reserved		Reserved													
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Register descriptions

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	Reserved		DELTA_T													
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

FCCU_DELTA_T field descriptions

Field	Description
0–1 Reserved	This field is reserved.
2–15 Reserved	This field is reserved.
16–17 Reserved	This field is reserved.
18–31 DELTA_T	Bistable Minimum Fault-Output (EOUT) Faulty Interval Applies only to Bistable fault-output mode (FOM). Controls the minimum amount of time (T_{min}) that the fault-output (EOUT) timer runs according to this equation: $T_{min} = 250 \mu s + DELTA\_T$ The following values and meanings are for DELTA_T: 00_0000_0000_0000: 0 μs 00_0000_0000_0001: 1 μs ... 11_1111_1111_1110: 16,382 μs 11_1111_1111_1111: 16,383 μs NOTE: The durations shown for the DELTA_T values are for CLKSAFE clock signals running at exactly 16 MHz. However, the CLKSAFE signals are sourced from an internal chip clock signal (for example, IRC)—or an integer division of it (see the chip-specific FCCU information)—that has a trimmed frequency variation. Therefore, the DELTA_T durations may vary. See your chip's data sheet for the trimmed frequency variation (for example, δF_{var}).

62.10.25 IRQ Alarm Enable (FCCU_IRQ_ALARM_ENn)

These registers enable the corresponding IRQ alarm.

IRQ Alarm Enable (FCCU_IRQ_ALARM_ENn) register (value of n)	Offset	IRQENx fields (value of x)	
		Most significant (leftmost) bit	Least significant (rightmost) bit
0	0FCh	31	0
1	100h	63	32

NOTE

These registers can be written only when the FCCU is in CONFIG state.

Address: 0h base + FCh offset + (4d × i), where i=0d to 1d

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R																																
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

FCCU_IRQ_ALARM_ENn field descriptions

Field	Description
0–31 IRQENx	IRQ alarm enable For the mapping of the IRQENx fields among registers, see the earlier table in this register description. 0 Alarm is disabled for error source x. 1 Alarm is enabled for error source x.

62.10.26 NMI Enable (FCCU_NMI_ENn)

These registers enable the NMI.

[Table 62-54](#) shows FCCU NMI enable register channels.

Table 62-54. FCCU NMI enable register channels

Address offset	Register name	Channel range (x) (bit location [0:31])
10Ch	FCCU_NMI_EN0	NMIENE[31:0]
110h	FCCU_NMI_EN1	NMIENE[63:32]

NOTE

These registers can be written only when the FCCU is in CONFIG state.

Address: 0h base + 10Ch offset + (4d × i), where i=0d to 1d

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R																																
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	

FCCU_NMI_ENn field descriptions

Field	Description
0–31 NMIENx	NMI enable See Table 62-54 for register offset to channel number relationship.

FCCU_NMI_ENn field descriptions (continued)

Field	Description
0	NMI is disabled for error (NCF) source x.
1	NMI is enabled for error (NCF) source x.

62.10.27 Bistable Fault-Output (EOUT) Mode Signaling Enable (FCCU_EOUT_SIG_ENn)

Writable only when FCCU is in Configuration state. Applies only when the EOUT signals are active ([FCCU_SET_AFTER_RESET](#)). When FCCU is configured for Bistable fault-output mode ([FOM](#)), controls whether fault-output (EOUT) signaling is enabled for the associated noncritical fault channel (x). (For other fault-output modes, fault-output signaling is always enabled, regardless of the value of this field.)

When FCCU is configured for a fault-output mode other than Bistable, controls whether FCCU asserts the FIF signal when a fault occurs on the associated noncritical fault channel (x). (When FCCU is configured for Bistable fault-output mode, FCCU always asserts the FIF signal when a fault occurs on the channel, regardless of the value of this field.)

Bistable Fault-Output (EOUT) Mode Signaling Enable (FCCU_EOUT_SIG_ENn) register (value of n)	Offset	EOUTENx fields (value of x)	
		Most significant (leftmost) bit	Least significant (rightmost) bit
0	11Ch	31	0
1	120h	63	32

Address: 0h base + 11Ch offset + (4d × i), where i=0d to 1d

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R																																
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

FCCU_EOUT_SIG_ENn field descriptions

Field	Description
0–31 EOUTENx	Bistable Fault-Output (EOUT) Mode Signaling Enable x Writable only when FCCU is in Configuration state. Applies only when the EOUT signals are active (FCCU_SET_AFTER_RESET). When FCCU is configured for Bistable fault-output mode (FOM), controls whether fault-output (EOUT) signaling is enabled for the associated noncritical fault channel (x). (For other fault-output modes, fault-output signaling is always enabled, regardless of the value of this field.)

FCCU_EOUT_SIG_ENn field descriptions (continued)

Field	Description
	When FCCU is configured for a fault-output mode other than Bistable, controls whether FCCU asserts the FIF signal when a fault occurs on the associated noncritical fault channel (x). (When FCCU is configured for Bistable fault-output mode, FCCU always asserts the FIF signal when a fault occurs on the channel, regardless of the value of this field.) NOTE: When FCCU is configured for a fault-output mode other than Bistable, you must set this field to enabled to ensure that FCCU asserts the FIF signal when a fault occurs on the associated noncritical fault channel (x) so the FOSU module doesn't mistakenly generate a chip reset. For the mapping of the EOUTENx fields among registers, see the earlier table in this register description. 0 Disabled 1 Enabled

62.10.28 Configuration registers

62.10.28.1 Definition

Configuration registers are registers that:

- Let you configure FCCU's Alarm-state timer interval, fault channels, and fault-output (EOUT) signals.
- You can write to only when the configuration is not locked (see [Lock and unlock the configuration](#)) and FCCU is in Configuration state (see [Put FCCU in Configuration or Normal state](#)).
- Save the values you write to them while FCCU is in Configuration state only after you manually put FCCU in Normal state. If FCCU automatically leaves Configuration state and enters Normal state because the configuration-timer interval (TO) expires (called a Configuration-state timeout), FCCU changes the value of the FCCU_CFG register to its Configuration-state-timeout value and the value of each of the other configuration registers to its reset value. For information on the Configuration-state timeout value, see [FCCU_CFG register bit value sources \(N and C\) by event](#).

62.10.28.2 The configuration registers

The following registers are the configuration registers, listed in offset order from lowest to highest:

- [Configuration \(FCCU_CFG\)](#)
- [Noncritical Fault Configuration \(FCCU_NCF_CFGn\)](#)
- [Noncritical Fault State Configuration \(FCCU_NCFS_CFGn\)](#)
- [Noncritical Fault Enable \(FCCU_NCF_En\)](#)

- Noncritical Fault Timeout Enable (FCCU_NCF_TOEn)
- Noncritical Fault Timeout (FCCU_NCF_TO)
- Delta T (FCCU_DELTA_T)
- IRQ Alarm Enable (FCCU_IRQ_ALARM_ENn)
- NMI Enable (FCCU_NMI_ENn)
- Bistable Fault-Output (EOUT) Mode Signaling Enable (FCCU_EOUT_SIG_ENn)

62.10.29 FCCU_CFG register bit value sources (N and C) by event

Certain events affect the bit values of the [Configuration \(FCCU_CFG\)](#) register; when one of these events occurs, the bit value is either unaffected, or FCCU changes it to one of the following:

- N: a chip-specific (or, for some chips, user-specified) value provided by the bit's associated NVMCFG signal
- C: a chip-specific value provided by the bit's associated constant inside the module

For each bit in the [Configuration \(FCCU_CFG\)](#) register, this table shows the associated NVMCFG signal, if any, and indicates the source of the bit value for the events that affect it. For specific values of *N* and *C* for each bit, see the chip-specific FCCU information.

Bit	NVMCFG signal	FCCU reset	Configuration-state timeout	OP31 operation
0	21	N	C	N
1	20	N	C	N
2	19	N	C	N
3	18	N	C	N
4	17	N	C	N
5	16	N	C	N
6	15	N	C	N
7	—	C	C	(Unaffected)
8	—	C	C	(Unaffected)
9	—	C	C	(Unaffected)
10	—	C	C	(Unaffected)
11	—	C	C	(Unaffected)
12	—	C	C	(Unaffected)
13	—	C	C	(Unaffected)
14	—	C	C	(Unaffected)
15	—	C	C	(Unaffected)

Table continues on the next page...

Bit	NVMCFG signal	FCCU reset	Configuration-state timeout	OP31 operation
16	14	N	C	N
17	13	N	C	N
18	12	N	C	N
19	—	C	C	(Unaffected)
20	11	N	C	N
21	10	N	C	N
22	9	N	C	N
23	8	N	C	N
24	7	N	C	N
25	6	N	C	N
26	5	N	C	N
27	4	N	C	N
28	3	N	C	N
29	2	N	C	N
30	1	N	C	N
31	0	N	C	N

62.11 FCCU Output Supervision Unit

The FOSU provides a supervision of the primary fault notification path by analyzing FCCU behavior for correctness. It waits for any reaction of the FCCU in a fixed time window after a fault is signaled.

The intention of the FOSU is to provide a secondary fault reaction path in most cases when the FCCU fails but not to needlessly propagate a fault which is already handled by the FCCU in a full chip reset. Only a failed primary fault reaction (that is, FCCU's failure) is a reason for the secondary reaction to take over (and generate a reset request).

There is a 'do nothing' input coming from the FCCU that indicates that the FCCU is programmed for no reaction for ALL FAULTS. It is a "static" input in the sense that it does not change after FCCU configuration. The FOSU masks the incoming faults with the 'do nothing' control from the FCCU, meaning that a fault is not captured by the FOSU if the 'do nothing' signal is asserted, (i.e., a disabled fault). There is no minimum pulse width requirement on the fault indication other than what is required by the technology, which is the same as that of the FCCU. FOSU does not monitor FCCU for the case of faults occurring during CONFIG state.

The FOSU contains a timer with a duration of FOSU_COUNT, driven by CLKSAFE. The timer is initialized and started on any captured, enabled fault. While the timer is running, any subsequent captured fault will neither restart nor reinitialize the timer. The timer is stopped when the FCCU shows any of the following reactions (the FOSU does not check whether the reaction is the configured one for the faults which occurred):

- Functional reset
- IRQ: NMI or Alarm
- Error out triggered (by FCCU or by SW)

When the timer is stopped, the fault capture logic is cleared in order to ensure that the timer is not restarted due to faults still 'stuck' in the capture logic. The timer will then be restarted by the next new failure indication. When the timer expires, the FOSU's failure indicator output is asserted after it ensures that the fault is enabled and the static "fccu program to do nothing" signal is deasserted. This is because FCCU uses settings after it exits CONFIG state, even if fault captured before the exit.

The FOSU's failure indication output is connected to a reset input of the system reset state machine and is treated like a power-on reset event. When this reset is asserted, the FOSU's capture logic is cleared, its timer is kept stopped and in a non-expired state, and its failure indicator output is deasserted.

NOTE

FOSU is triggered on assertion of enabled fault. In case the triggering fault is disabled, FOSU times-out without reaction.

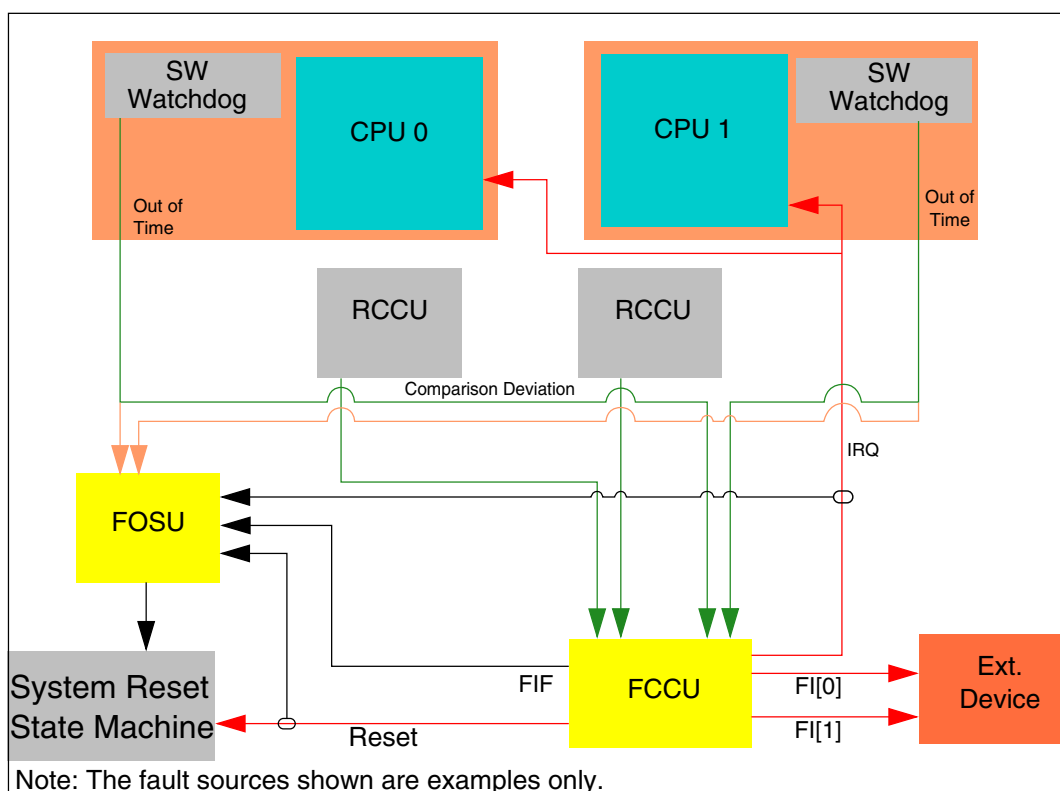


Figure 62-56. FOSU connections to the FCCU and system reset state machine

It is important to remember that there will be no FCCU reaction to a fault while the FCCU is in the CONFIG state. For this reason, the FCCU should not be kept in CONFIG for longer than the FOSU_COUNT duration. Otherwise, there is a risk that an incoming error report causing the FOSU to mistakenly see the FCCU as having failed, and then resets the MCU.

NOTE

FCCU and FOSU timeout counter settings: The FCCU counter value should always be programmed less than the FOSU counter value, FOSU_COUNT, or resets may be generated by an FOSU time-out. This ensures that the FOSU reacts only when the FCCU fails to react to any particular fault.

62.12 Use cases and limitations

62.12.1 Configuration guidelines

Follow these guidelines to configure FCCU:

- If you want FCCU to react to a fault on a noncritical fault channel:
 - Enable the channel ([NCFEx](#)).
 - Enable at least one type of Fault-state reaction for the channel: chip reset ([NCFSCx](#)), fault-output (EOUT) signaling ([EOUTENx](#)), or nonmaskable interrupt ([NMIENx](#)).
 - If you enable chip reset as the type of Fault-state reaction for the channel ([NCFSCx](#)), enable either Alarm state ([NCFTOEx](#)) or at least one other type of Fault-state reaction for the channel: fault-output (EOUT) signaling ([EOUTENx](#)) or nonmaskable interrupt ([NMIENx](#)).
 - If you enable Alarm state for the channel ([NCFTOEx](#)), enable the Alarm-state reaction ([IRQENx](#)).
 - If you enable Alarm state for the channel ([NCFTOEx](#)), make sure the Alarm-state timer interval ([TO](#)) is less than the FOSU module's timer interval; otherwise, FOSU generates a chip reset every time a fault occurs on the channel. The FOSU timer interval (FOSU_COUNT) is chip-specific. See the chip-specific FCCU information.

NOTE

If you enable a noncritical fault channel but disable all reactions for that channel, FCCU changes state if necessary but doesn't perform any reaction because reactions are disabled. If you enable reactions for a noncritical fault channel but disable that channel, and FCCU is in Normal state when a fault occurs on the channel, FCCU doesn't enter Alarm or Fault state and therefore doesn't perform any reaction.

62.12.2 Recommendations to configure FCCU

1. After a reset cycle in which the FOSU, FCCU, and system are initialized (such as caused by power-up, RESET pin, FOSU reset, or STCU reset), the following steps could be followed to configure FCCU:
 - a. Check and clear any pending fault status
 - b. Verify FCCU is in NORMAL state, else repeat step(a) above
 - c. Configure FCCU
2. After a reset cycle in which the system is re-initialized but the FOSU and FCCU are not re-initialized, the following steps could be followed to reconfigure FCCU:
 - a. If active, wait for the Error out T_min to expire
 - b. Check and clear fault status
 - c. Error pin moves to "non faulty" state, once fault status is cleared and T_min expires

- d. Verify FCCU is in NORMAL state, else repeat step(a) above
- e. Read and verify value in FCCU_NCF_Ex
- f. Reconfigure FCCU, if necessary

Chapter 63

Self-Test Control Unit (STCU2)

63.1 Chip-specific STCU information

The STCU controls the execution of built-in self-tests (BISTs) of the chip.

The STCU manages two primary types of BISTs:

- MBIST: memory (SRAM/ROM) BIST
- LBIST: logic BIST for digital logic

The STCU has two sets of conditions under which it applies a self-test sequence:

- Off-line condition: After the user stores self-test parameters in flash memory and a reset cycle is initiated by a power-up event, RESET pin assertion, or FOSU reset, the STCU loads the stored parameters, executes a self-test, and then resets the chip.
- On-line condition: After the user initiates one or more self-tests during normal chip operation, the STCU executes the self-tests and then resets the chip.

The STCU controls the clocking and the reset of the chip during a self-test.

NOTE

If both MBIST and LBIST are scheduled to be executed, then MBIST should be executed first. LBIST can be executed only after MBIST is completed.

63.1.1 Supported BIST sequences

NXP has validated and therefore supports only particular STCU2 built-in self-test (BIST) sequences for this chip. For descriptions of the supported BIST sequences for this chip, see the Application Note titled *MPC5777C STCU Quick Start Guide* (document AN5288).

63.1.2 Configure or bypass STCU2 off-line self-tests

During reset, the STCU2 has a watchdog timer that can time out and delay exit from reset by at least ~8 ms. To prevent this delay, program DCF records either to configure and run or to bypass STCU2 off-line self-tests at startup.

After programming one set of DCF records either to run or bypass off-line self-tests, do not program subsequent DCF records to change the initial selection of run or bypass.

For information about configuring and running self-tests, see [Supported BIST sequences](#).

To bypass STCU2 off-line self-test operation:

1. Program a DCF record for the client STCU_SKC (see [Table 5-6](#)) to write the Key1 and Key2 codes for off-line self-test. See [STCU2 SK Code Register \(STCU2_SKC\)](#) for details about the register.
2. Program a DCF record for the client STCU_RUN (see [Table 5-6](#)) to write 1 to STCU2_RUN[BYP]. See [STCU2 Run Register \(STCU2_RUN\)](#) for details about STCU2_RUN[BYP].

63.1.3 STCU interfaces to other modules

The STCU interfaces with other modules on the chip, including:

- SSCM: For off-line self-tests, the STCU uses the SSCM DCF bus to load self-test parameters from DCF records in flash memory. After the reset that precedes the off-line self-test, this SSCM interface is able to write the configuration parameters and start the off-line self-test execution only one time.
- IPS bus (Peripheral Bridge):
 - For on-line self-tests, the CPU accesses STCU memory-mapped registers to initiate execution of the self-tests.
 - For both off-line and on-line self-tests, the CPU accesses STCU memory-mapped registers to check self-test results.
- FCCU: For off-line self-tests, the STCU can be configured through DCF records to classify a hardware fault as a critical fault (CF) or non-critical fault (NCF).

- If a subsequent off-line self-test detects a hardware fault that is classified as a NCF, the FCCU's default response is no reaction.
- If a subsequent off-line self-test detects a hardware fault that is classified as a CF, the FCCU responds to the CF by initiating a new reset cycle.

NOTE

On this chip, the critical fault (CF) output of the STCU is connected to a non-critical fault (NCF) input of the FCCU. This input is programmed to issue a reset by default. The FCCU itself is not configured to receive "native" CF inputs.

63.1.4 On-line LBIST and MBIST clock configuration

Before running an on-line LBIST or MBIST, configure clocking as follows to ensure proper execution:

1. PLL0 is the global clock source: Set PLL0 dividers to generate 200 MHz at PLL0's output.
2. Configure PLL0's clock source to be the crystal oscillator (XOSC): Set SIU_SYSDIV[PLL0SEL] to 0.
3. Select the non-FM PLL, PLL0, to drive the system clock: set SIU_SYSDIV[SYSCCLKSEL] to 11b.
4. Disable the Progressive Clock Switch feature: Set SIU_SYSDIV[PCSEN] to 0.
5. Set the System Clock Divider to divide by 1: Set SIU_SYSDIV[SYSCCLKDIV] to 1xx.
6. Set the Peripheral Clock Select: Set SIU_SYSDIV[PERCLKSEL] to 0.
7. Ensure the fm_per_clk divider is 2: Ensure SIU_SYSDIV[FMPPERDIV] is 00b.
8. Set the LFAST clock source to be the peripheral clock coming from PLL: Set SIU_LFCLKCFG[LFCLKSEL] to 10b.
9. Set SIU_SYSDIV[MCANSEL] to 1.
10. Set FlexCAN's CAN_CTRL1[CLKSRC] to 1.
11. Set SWT_A's SWT_CR[CSL] to 0 and SWT_B's SWT_CR[CSL] to 0.

In addition, before executing an on-line MBIST, set the following clock divider values:

1. Set the eTPU clock divider to 2: Set SIU_SYSDIV[ETPUDIV] to 0.
2. Set the TCK divider to 3: Set STCU2_CFG[CLK_CFG] to 010b.

In addition, before executing an on-line LBIST:

- Configure the shift speed clock for the applicable LBIST with a divider equal to or greater than 2: Set the applicable STCU2_LB_CTRL n [SHS] field to a value of 001b or greater.

During an on-line LBIST, clock dividers are bypassed by default.

63.1.5 STCU2_PLL_CFG configuration of PLLDIG_PLL0DV

When used, the STCU2's [PLL interface](#) enables the STCU2 module to control PLL0 during off-line self-test operations. In this case, the values programmed into the fields of the [STCU2 PLL Configuration Register \(STCU2_PLL_CFG\)](#) configure the corresponding fields of the [PLLDIG PLL0 Divider Register \(PLLDIG_PLL0DV\)](#).

The following table shows the correlation between STCU2_PLL_CFG fields and PLLDIG_PLL0DV fields.

Table 63-1. STCU2_PLL_CFG and PLLDIG_PLL0DV field alignment

STCU2_PLL_CFG field	PLLDIG_PLL0DV field
PLLLDF	MFD
PLLIDF	PREDIV
PLLODF	RFDPHI

63.1.6 Wait time for writing to the on-line registers

On this chip, no additional wait time applies to writing the on-line registers.

63.1.7 On-line MBIST

After the STCU2 triggers the execution of an on-line MBIST, the user cannot perform any access until the MBIST ends.

At the end of the execution of an on-line MBIST, the STCU2 does not generate any reset automatically. The reason is that the software application must manage the consequences of write operations that the MBIST's execution performs.

After the execution of an on-line MBIST ends, the user must ensure that memory controllers are in the correct state.

- Reset the chip to clear the memory output.
- If any memory content is destroyed by an MBIST, reload the content after the MBIST.

It is recommended to enable MBIST interrupts only when executing MBISTs exclusively, not MBISTs combined with LBISTs. If an MBIST is combined with an LBIST in the same test run, the LBIST triggers a reset after the test execution, and the MBIST interrupt will be pending coming out of reset.

This chip implements 65 MBIST partitions.

Table 63-2. MBIST partitions

MBIST partition	Memory
MBIST0	SRAM 1_4
MBIST1	SRAM 1_3
MBIST2	SRAM 1_2
MBIST3	SRAM 1_1
MBIST4	SRAM 2_4
MBIST5	SRAM 2_3
MBIST6	SRAM 2_2
MBIST7	SRAM 2_1
MBIST8	FEC FIFO RAM
MBIST9	FEC MIB RAM
MBIST10	DMA RAM 2
MBIST11	DMA RAM 1
MBIST12	eTPU N3 SCM RAM 2
MBIST13	eTPU N3 SCM RAM 1
MBIST14	eTPU S32D SCM RAM
MBIST15	eTPU N3 SPM RAM 3
MBIST16	eTPU N3 SPM RAM 2
MBIST17	eTPU N3 SPM RAM 1
MBIST18	eTPU S32D SPM RAM 2
MBIST19	eTPU S32D SPM RAM 1
MBIST20	CSE RAM
MBIST21	Core1 cache data RAM 1_4

Table continues on the next page...

Table 63-2. MBIST partitions (continued)

MBIST partition	Memory
MBIST22	Core1 cache data RAM 1_3
MBIST23	Core1 cache data RAM 1_2
MBIST24	Core1 cache data RAM 1_1
MBIST25	Core1 cache data RAM 2_4
MBIST26	Core1 cache data RAM 2_3
MBIST27	Core1 cache data RAM 2_2
MBIST28	Core1 cache data RAM 2_1
MBIST29	Core0 cache data RAM 1_4
MBIST30	Core0 cache data RAM 1_3
MBIST31	Core0 cache data RAM 1_2
MBIST32	Core0 cache data RAM 1_1
MBIST33	Core0 cache data RAM 2_4
MBIST34	Core0 cache data RAM 2_3
MBIST35	Core0 cache data RAM 2_2
MBIST36	Core0 cache data RAM 2_1
MBIST37	Core1 cache tag RAM 1_4
MBIST38	Core1 cache tag RAM 1_3
MBIST39	Core1 cache tag RAM 1_2
MBIST40	Core1 cache tag RAM 1_1
MBIST41	Core1 cache tag RAM 2_4
MBIST42	Core1 cache tag RAM 2_3
MBIST43	Core1 cache tag RAM 2_2
MBIST44	Core1 cache tag RAM 2_1
MBIST45	Core0 cache tag RAM 1_4
MBIST46	Core0 cache tag RAM 1_3
MBIST47	Core0 cache tag RAM 1_2
MBIST48	Core0 cache tag RAM 1_1
MBIST49	Core0 cache tag RAM 2_4
MBIST50	Core0 cache tag RAM 2_3
MBIST51	Core0 cache tag RAM 2_2
MBIST52	Core0 cache tag RAM 2_1
MBIST53	Core1 cache "dirty" RAM
MBIST54	Core0 cache "dirty" RAM
MBIST55	FlexCAN RAM 1_2
MBIST56	FlexCAN RAM 1_1
MBIST57	FlexCAN RAM 2_2
MBIST58	FlexCAN RAM 2_1
MBIST59	M_CAN RAM
MBIST60	eTPU Nexus RAM 1_2

Table continues on the next page...

Table 63-2. MBIST partitions (continued)

MBIST partition	Memory
MBIST61	eTPU Nexus RAM 1_1
MBIST62	eTPU Nexus RAM 2_2
MBIST63	eTPU Nexus RAM 2_1
MBIST64	BAM ROM

63.1.8 LBIST scenarios

This chip implements six logical partitions, and each one has an LBIST Controller. The STCU controls the LBIST execution of all partitions.

Table 63-3. LBIST partition mapping

LBIST partition	Modules
LBIST0	Core1
LBIST1	Core0
LBIST2	Checker core, RCCU
LBIST3	Peripheral Bridge B (AIPS_0) and associated peripherals: • SDADC_1 and SDADC_3 • M_CAN Subsystem • CRC • DECFILTER_A to DECFILTER_L • LFAST • DSPI_A, DSPI_B, and DSPI_C • eMIOS_0 • EQADC_A • eSCI_A and eSCI_B • eTPU_C • FCCU including FOSU • FlexCAN_0 and FlexCAN_1 • IGF_0 • PSI5_0 • REACM2 • SRX_0 • SIPI

Table continues on the next page...

Table 63-3. LBIST partition mapping (continued)

LBIST partition	Modules
LBIST4	Peripheral Bridge A (AIPS_1) and associated peripherals: • SDADC_2 and SDADC_4 • CMU_0 to CMU_8 • DSPI_D and DSPI_E • EBI • eMIOS_1 • EQADC_B • eSCI_D, eSCI_E, and eSCI_F • eTPU_A/B • FCCU including FOSU • FlexCAN_2 and FlexCAN_3 • IGF_1 • PIT_RTI • PSI5_1 • SRX_1
LBIST5	Core platform modules: • XBAR • EIM • ERM • PCU • MPU_0 • MPU_1 • CSE • FEC • PFLASH • INTC • PRAMC_0 • PRAMC_1 • eDMA_A • eDMA_B • STM • SWT_A • SWT_B • C55FMC flash memory array

63.1.8.1 Off-line LBIST

The chip supports off-line LBIST operation for all six LBIST partitions.

Summary of an off-line LBIST sequence:

1. The user stores off-line self-test parameters in a DCF record in flash memory. The self-test can include multiple LBIST partitions.
 - For off-line self-tests, LBIST execution on individual LBIST partitions occurs serially. The chip does not support LBIST execution on individual LBIST partitions in parallel.

- An off-line LBIST can run with either the 16 MHz IRC or with PLL0.
 - For an off-line self-test, the STCU can enable and program PLL0 using STCU DCFs.
2. The user resets the chip by cycling power or asserting the RESET pin.
 3. The STCU loads the stored off-line LBIST parameters from the DCF record via the SSCM interface.
 4. The off-line LBIST executes and ends.
 5. Hardware issues a functional reset. The STCU is not affected by this reset.
 6. The user verifies the FCCU channels corresponding to STCU faults. If any fault is flagged, the detailed cause can be checked by reading STCU status registers.

63.1.8.2 On-line LBIST

The chip supports on-line LBIST operation on all six LBIST partitions.

NOTE

When LBIST execution begins, the embedded flash memory must be in an idle state; program or erase operations must not be active. See the [Functional Description](#) of the embedded flash memory for details about its states.

Summary of an on-line LBIST sequence:

1. The user initiates an on-line LBIST by programming STCU memory-mapped registers.
 - An on-line LBIST is able to test only one partition at a time.
 - An on-line LBIST can run with the functional operating frequencies. The maximum-frequency clock for the LBIST partition is derived from PLL0.
 - The user can initiate a sequence of more than one on-line LBIST. Multiple on-line LBISTs can execute serially, with each self-test run on an individual partition.
 - Program the LBRMSW register to generate a global functional reset after an on-line LBIST. Avoid using a dedicated reset due to random data injected by the LBIST Controller.
2. Hardware forces all LBIST partitions into a quiescent state.

3. The on-line LBIST executes and ends.
4. After the completion of the LBIST on-line self-test sequence, hardware issues a functional reset. The STCU is not affected by this reset.
5. The user verifies the FCCU channels corresponding to STCU faults. If any fault is flagged, the detailed cause can be checked by reading STCU status registers.

For on-line LBISTs, this chip does not support the following features:

- Interrupt generation after each LBIST on an individual partition
- Software abort during an on-line LBIST

NOTE

An abort does occur under these conditions:

- A reset cycle is triggered in the middle of a self-test execution.
- The reset source is not POR, RESET pin, FOSU, or the STCU itself.

A reset caused by POR, RESET pin, or FOSU does not cause an abort because it resets the STCU and causes it to run the self-test again. A reset caused by the STCU itself implies successful completion of the self-test procedure.

63.1.8.3 Diagnostic self-test: On-line LBIST from JTAG/Nexus interface

A "diagnostic self-test" is an on-line LBIST that is invoked via the chip's JTAG/Nexus debug/development interface. Via the chip's JTAG pins, a JTAG tool can access the Nexus debug interface in the CPU and use it to configure the STCU without interfering with normal chip operation.

However, after the LBIST of a partition starts, the STCU takes control of the chip, putting all partitions that are not being tested in a quiescent state. The chip resets at the end of the LBIST.

63.1.8.4 Multiple serial LBISTs

The chip allows serial execution of off-line and on-line LBISTs. For example, this sequence is valid:

1. The user programs a DCF record in the flash memory to perform an off-line LBIST.
2. A power-on reset (POR) occurs.
3. The STCU executes the off-line self-test.
4. A functional reset occurs.
5. The CPU boots. The user programs STCU registers using the CPU to execute an on-line LBIST. An on-line LBIST is initiated.
6. A functional reset occurs.
7. The CPU boots again. No off-line or on-line self-test is initiated.

63.1.8.5 LBIST execution and dual cores

On this dual-core chip, either core can program (schedule) execution of an LBIST. After the LBIST of a partition begins, both cores and all other logic not included in that partition are maintained in a reset state. Neither core can operate during the LBIST execution of any partition, regardless of which core initially configured the LBIST.

63.1.9 STCU register default values

The following table documents STCU register default values that are specific to this chip. For the default values of all other STCU registers, see [Register description](#).

NOTE

Some of these registers are DCF clients whose values are overwritten by the values in STCU2-specific DCF records in UTEST flash memory. If the user programs a value for one of these DCF records, the corresponding register reflects the user-programmed value.

Table 63-4. STCU register default values

Register	Default value
STCU2_CFG	0000_0000h
STCU2_PLL_CFG	0000_0000h

Table continues on the next page...

Table 63-4. STCU register default values (continued)

Register	Default value
STCU2_WDG	0000_FFFFh
STCU2_ERR_FM	0000_0000h
STCU2_LBRMSW	0000_003Fh
STCU2_LBUFM	0000_0000h
STCU2_MBUFML, STCU2_MBUFMM, STCU2_MBUFMH	0000_0000h
STCU2_LB_CTRL n	0000_0000h
STCU2_LB_PCS n	0000_0000h
STCU2_LB_MISREL n , STCU2_LB_MISREH n	FFFF_FFFFh
STCU2_LB_MISRRL n , STCU2_LB_MISRRH n	0000_0000h
STCU2_LB_MISRELSW n , STCU2_LB_MISREHSW n	FFFF_FFFFh
STCU2_LB_MISRRLSW n , STCU2_LB_MISRRLHSW n	0000_0000h
STCU2_MB_CTRL n	0000_0000h

63.1.10 AUTOLOCK_VALUE for register write access and DCF_COMPLETION value for DCF write completion

[Register write-access watchdog timer](#) refers to two clock-cycle values related to the STCU2's write-access watchdog. The following table provides these values on this chip.

Clock-cycle value	Number of STCU2 clock cycles
AUTOLOCK_VALUE	1024
DCF_COMPLETION	16

63.2 Introduction

Self Test Control Unit (STCU2) is a comprehensive programmable hardware module that controls the self test sequence applied both during the Off-Line and/or On-Line conditions. It is able to manage by hardware the device's Logic Built-In Self Test (LBIST) and the SRAM/ROM Built-In Self Test (MBIST) blocks. The STCU2 includes the SSCM DCF bus to load the Self-test parameters (L/MBIST scheduling activity, LBIST setup, Unrecoverable/Recoverable faults management, and MISR expected values, PLL management, and so on) from flash memory during the Off-Line Self-Test phase. This interface is able only to write the configuration parameters and start the Self-

Test execution once after the STCU2 global reset has been applied. The register access by software is granted by an IPS interface with the purpose to check the results of the Off-Line Self-Test but also to load/check the execution of the On-Line Self-Test.

The STCU2 includes:

- A programmable Watchdog timer (WDG) to check the Self-Test operations (both LBIST and MBIST) have been completed within the assigned time slot or the STCU2 RUN or BYPASS bits have been programmed before Watchdog time-out
- The PLL direct control during the Off-Line Self-Test sequence
- Interrupt lines enabled only during the On-Line Self-Test to simplify the software handling of Self-Test execution end
- Two sets of Result registers to collect the L/MBIST result during Off/On-Line Self-Test separately

To increase the security capability of this module, a different Security Key sequence is also required during Off/On-Line Self-Test in order to unlock the write access to the STCU2 registers. The STCU2 also implements two power-saving mechanisms to switch off:

- The global STCU2 clock after Self-Test sequence is completed
- The register ITF clock when the WDG time-out is detected after a Double Security Key sequence is applied during the application run (available only when IPS write access is enabled otherwise the clock is off).

63.3 Main features

- Double registers interface (SSCM DCF with priority over IPS)
- SSCM DCF write one time register interface
- IPS read and write (depends on SSCM enable parameters and STCU2_CFG.WRP bit) register interface
- Programmable scheduler for LBIST/MBIST execution
- LBIST concurrent/sequential execution
- MBIST concurrent/sequential execution
- Programmable LBIST delayed concurrent start

- Programmable Internal clock prescaler to reduce Internal and MBIST/LBIST TCK clocks
- Programmable PLL direct control during Off-Line Self-test sequence
- PLL Lock signal monitoring during Off/On-Line Self Test
- MBIST dedicated On-Line interrupt source generation
- Programmable WDG Timer for Self-Check M/LBIST Time Execution
- Hard Coded WDG time-out to flag off-line initialization faults
- Hard Coded WDG time-out to Auto-lock STCU2 write access after Double-Key sequence
- LBIST and SRAM/ROM MBIST Off-Line status flags
- STCU2 internal errors Off-Line status flag
- LBIST and SRAM/ROM MBIST On-Line status flags
- STCU2 internal errors On-Line Status flag
- LBIST and SRAM/ROM MBIST Off/On-Line programmable failure severity - unrecoverable fault/recoverable fault (UF/RF)
- STCU2 internal errors Off-On/Line programmable failure severity - unrecoverable fault/recoverable fault (UF/RF)
- STCU2 errors (UF/RF) lines for FCCU signaling
- Redundant UF/RF generation logic to improve reliability
- FCCU UF/RF faults injection mechanism
- LBIST MISRE registers access to perform redundant check by software
- Self-Test Bypass capability
- Global write register's protection mechanism based on Double Security Key
- STCU2 Global Auto Power-saving when Self-Test is completed, bypass mode is selected or WDT time-out is detected
- STCU2 ITF/WDG clock wake-up mechanism when software application writes Double Security Key sequence (available when STCU2_CFG.WRP bit is active)
- STCU2 ITF/WDG Auto Power-saving when Hard Coded WDG time-out is detected after software application writes Double Security Key sequence (available when STCU2_CFG.WRP bit is active)

63.4 Block diagram

The top level diagram of the STCU2 module is given in [Figure 63-1](#).

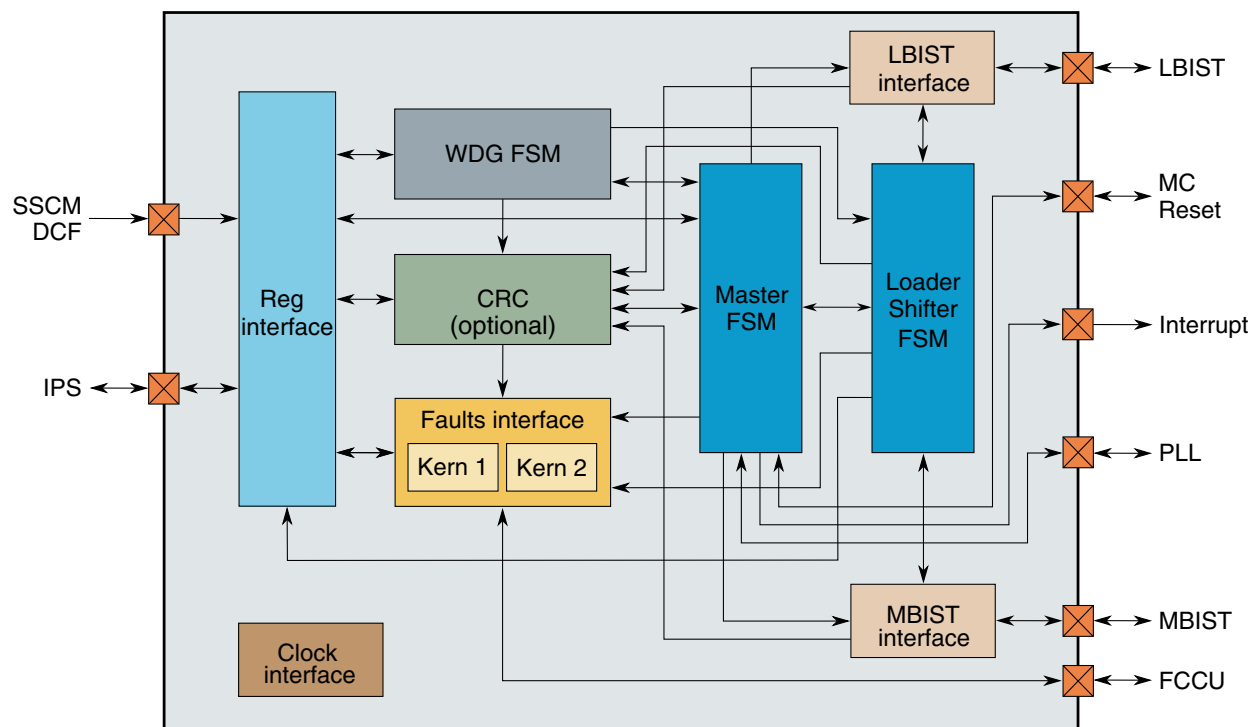


Figure 63-1. STCU2 block diagram

The STCU2 module includes the following sub-modules:

- **Reg interface:** This module includes the registers, the security key logic, the IPS and SSCM DCF bus interface
- **Fault interface:** This module collects the fault conditions related to L/MBIST execution and STCU2 internal failures and, depending on the UF/RF configuration of each one of them, sets the global STCU2 UF or RF flag. It also manages the UF/RF fault lines and the Set/Clear injection mechanism provided by the FCCU. To improve the intrinsic reliability of this critical logic, the generation logic is duplicated.
- **Clock Interface:** This module manages the internal and the L/MBIST TCK clock prescaler, the internal Clock Gating Power Saving and the wake-up clock feature.
- **WDG FSM:** This module includes two different Watchdogs. The first one is Hard Coded and is used to Auto Lock the STCU2 access forcing a reset condition on the Double Security Key registers while the second one has a double functionality. After

a reset event initializes the STCU2, it is used as Hard Coded Watchdog time-out while during L/MBIST run it is used as programmable Watchdog timer to check the L/MBIST have been completed in the assigned L/MBIST time slot.

- **STCU2 Master FSM:** This module includes the main FSM of the STCU2 used to coordinate and schedule all the operations performed during the Self-Test Sequence.
- **STCU2 Load Shifter FSM:** This module includes the common shifter register and the related state machine used to program the L/MBIST registers and read back the data to be checked at the end of each Test operation.
- **LBIST interface:** This module includes the interface between the device's LBIST engines and the STCU2 controllers.
- **MBIST interface:** This module includes the interface between the MBIST controller and STCU2 controllers.

63.5 IPS bus interface

The IPS bus interface is a slave bus used for configuration purposes via CPU. The following bus read operations (contiguous byte enables) are supported:

- Word (32 bits) data read operations to any registers
- Low and high half-words (16 bits, data[31:16] or data[15:0]) data read operations to any registers
- Byte(8 bits, data[31:24] or data[23:16] or data[15:8] or data[7:0]) data read operations to any registers
- Any other operation (free byte enables or other operations) has to be avoided.

The same operations are also supported in write mode only in the following scenario:

- The Self-Test sequence parameters are loaded before Off-Line Self-test is executed
- The IPS access on a specific register is allowed by software only when the bit STCU2_CFG.WRP is cleared
- In all the other conditions the write operations are not supported and all the STCU2 registers can be only read. Only exceptions are the following read/write bits: STCU2_CFG.WRP, STCU2_ERR_STAT.UFSF and STCU2_ERR_STAT.RFSF. See related register description for additional details

The STCU2 module generates a transfer error in the following cases:

- Any write/read access to the register addresses not mapped on the peripheral but included in the address space of the peripheral
- Any write/read operation different from byte/halfword/word (free byte enables or other operations) on each register
- Any write operation on Double Security Key register applying a wrong sequence of keys (the two write operations cannot be interleaved with other access to STCU2 registers)
- Any write operation performed on a register when the Double Security keys have not been applied
- Any write operation performed on registers when the STCU2_CFG.WRP bit is set and the access is performed through software (IPS interface)
- Any write operation performed through software (IPS interface) on Read/On-Line_Write/Off-Line registers
- Any write operation performed off-line on registers in which writing operation is allowed only in the On-Line Self-Test phase when STCU2_CFG[WRP] = 1
- Any write operation performed on Read Only registers

The registers of the STCU2 module are accessible (read/write) in each access mode: user, supervisor or test.

In case there are write operations on bits marked as reserved, the transfer error is not generated.

NOTE

See the chip-specific STCU2 information for the wait time necessary to write to the online registers.

63.6 BISTs and BIST partitions

63.6.1 Definition: BIST

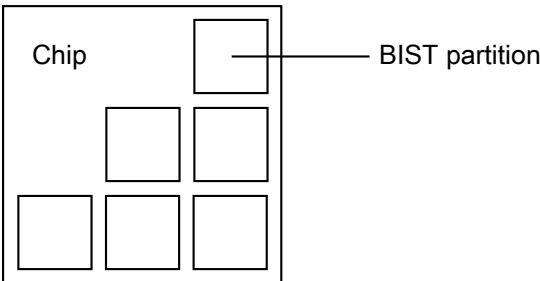
A *built-in self-test (BIST)* is a test that the chip can execute to verify the functional integrity of a part of itself. The chip uses STCU2 and other on-chip hardware to execute a BIST.

63.6.2 Definition: BIST partition

A *built-in self-test (BIST) partition* is a part of a chip for which a BIST has been defined. The hardware that is included in a given BIST partition is chip-specific. For a list of the hardware included in each of the BIST partitions on this chip, see the chip-specific STCU2 information.

63.6.3 Example: BIST partitions on a chip

This is an example of a chip with six built-in self-test (BIST) partitions:



63.6.4 Types of BIST partitions

The STCU2 module on this chip supports the following type or types of built-in self-test (BIST) partitions:

BIST partition type	Description
Memory BIST (MBIST) partition	An SRAM or ROM block
Logic BIST (LBIST) partition	One or more digital modules

63.7 BIST sequences

63.7.1 Definition: BIST sequence

A *built-in self-test (BIST) sequence* is a programmable series of one or more phases, each of which executes one or more individual BISTs.

63.7.2 STCU2 executes MBISTs before LBISTs

If a built-in self-test (BIST) sequence includes both MBISTs and LBISTs, STCU2 executes the MBISTs first.

63.7.3 Example: Single-phase BIST sequence

This is an example of a single-phase BIST sequence in which STCU2 executes only one BIST:

Phase	BIST executed
0	MBIST 16

63.7.4 Example: Multiphase BIST sequence

This is an example of a multiphase BIST sequence in which STCU2 executes more than one BIST in parallel in some phases:

Phase	BISTs executed
0	MBISTs 7, 38
1	MBIST 23
2	MBISTs 6, 16, 29, 23
3	LBIST 12
4	LBISTs 13, 8, 11
5	LBISTs 19, 6

63.7.5 Types of BIST sequences

The STCU2 module on this chip supports these types of built-in self-test (BIST) sequences:

BIST sequence type	Description
Online BIST sequence	A BIST sequence that STCU2 executes during runtime at the request of software. Application software configures and initiates execution of the BIST sequence by loading values into STCU2 registers. Use this type of BIST sequence when you want to test the chip hardware but need to minimize the system startup time. Also use it for failure diagnostics and quality control within a manufacturing environment. You can choose the most appropriate point during runtime at which to execute this type of BIST sequence.
Offline BIST sequence	A BIST sequence that STCU2 automatically executes during reset, either when power is first applied to the chip or when the chip resets STCU2. The chip's SSCM module configures and initiates execution of the BIST sequence by loading values into STCU2 registers from Device Configuration Format (DCF) records that are stored in the chip's UTEST flash memory. Use this type of BIST sequence when you want to test the chip hardware before runtime. The location of the DCF record area is chip-specific. The size of the DCF record area is sequence-specific. For the location of the DCF record area, see the chip-specific STCU2 information.

63.7.6 Supported BIST sequences

NXP has validated and therefore supports only specific built-in self-test (BIST) sequences for this chip. For a list of the supported BIST sequences, see the chip-specific STCU2 information.

63.7.7 Definition: Default offline BIST sequence

The *default offline built-in self-test (BIST) sequence* is the offline BIST sequence whose configuration NXP has programmed into the chip's Device Configuration Format (DCF) records that are stored in the chip's UTEST flash memory.

If you choose to use the default offline BIST sequence, or if it's the only one supported by NXP, there is no need to create additional DCF records for the offline BIST sequence.

63.8 Functional description

63.8.1 FSM description

The module has three state machines that work together: Master State Machine, Loader/Shifter State Machine, and the Watchdog State Machine. Basically the Master State Machine is the core unit of the STCU2 module. It coordinates all the Self Test operations and the other State Machines. The Loader/Shifter State Machine is used to program the MBIST and the LBIST parameters and to retrieve the related results depending on the parameters stored into the STCU2 registers and under the control of the Master State Machine. The Watchdog State Machine evaluates all the schedule time for MBIST and LBIST and the time-out in case of wrong STCU2 programming.

63.8.2 Reset management

Assertion of the STCU2 reset input signal initializes the STCU2 module status, forcing the Off-Line Self-Test condition and cleaning up all the registers and state machine of the module.

The reset signals generated by the STCU2 module at the end of the Self-Test execution phase depends on the Self-Test conditions and on the content of the LBRMSW register.

63.8.2.1 Off-Line reset generation

At the end of the Off-Line Self-Test execution the STCU2 raises the global functional reset.

63.8.2.2 On-Line reset generation

See the chip-specific STCU2 information for details about On-Line reset generation.

63.8.3 Built-in self-test scheduling

The STCU2 module has been designed to be very flexible allowing to program the parallel/serial execution of the MBIST or LBIST depending on the power/timing/coverage constraints. The limitation in the programming flexibility is described in [Design implementation information](#).

The mechanism used to provide this flexibility is a linked list where the starting pointer is the bitfield STCU2_CFG.PTR. The first LBIST is mapped on 00h, the second on 01h, and so on. The first MBIST is mapped to the address 10h, the second to 11h and so on.

The additional pointers are in the bitfield STCU2_LB_CTRL[X].PTR for LBIST[X] (where X is the selected LBIST) and STCU2_MB_CTRL[Y].PTR for MBIST (where Y is the selected MBIST) and have to be filled depending on the selected sequence of run. The additional bitfield STCU2_LB_CTRL.CSM and STCU2_MB_CTRL.CSM provide the flexibility to run concurrently or sequentially the chosen set of the LBIST or MBIST or to close the linked list setting the NIL pointer. For more details, see [STCU2 LBIST Control \(STCU2_LB_CTRL_n\)](#) and [STCU2 MBIST Control Register \(STCU2_MB_CTRL_n\)](#).

NOTE

Program operation should ensure that no flash write/erase operations are ongoing when online LBIST is triggered. Otherwise, a flash segment may be corrupted.

63.8.4 PLL interface

The STCU2 module is able to directly control the PLL during the Off-Line Self-Test operations depending on the status of the STCU2_RUN[MBPLEN] and STCU2_RUN[LBPLEN].

When STCU2_RUN[RUN] is set to start the Off-Line Self-test operations, the STCU2 takes control of the PLL and changes the PLL clock configurations parameters according to the values programmed into the STCU2_PLL_CFG register.

When the STCU2_RUN[MBPLEN] is active and the MBIST is currently selected, the STCU2 waits for the PLL lock signal. As soon as the lock happens, the STCU2 forces the clock to switch to the PLL output clock. This allows the MBIST to proceed at the PLL output frequency. At the end of the MBIST run the clock source is reverted to the original one.

Same behavior will be applied when STCU2_RUN[LBPLEN] is active and LBIST is currently selected.

63.8.5 Interrupt interface

Interrupt source:

- Triggered by MBIST (enabled by programming STCU2_RUNSW[MBIE] = 1)

As described in the [STCU2 Run Software Register \(STCU2_RUNSW\)](#) description, the interrupts are generated at the end of any concurrent run. This means that in case of multiple sequential run in the same session, the number of interrupts generated is equivalent to the number of the sequential session providing in such a way a fine granularity with respect to the completed run.

The MBIFLG bit in the [STCU2 Interrupt Flag Register \(STCU2_INT_FLG\)](#) register flag interrupt conditions and allow the clearing of interrupt requests. Interrupt bits are cleared by writing the related flag bit to 1 to enable the next interrupt request.

63.8.6 FCCU interface

The FCCU interface is the hardware flag mechanism towards the system, to indicate the occurrence of an Unrecoverable fault and/or a Recoverable fault failure during the Self Test sequence.

To diagnose physical defects on the two fault signals, a fault injection mechanism is also provided. In this case, the FCCU interface allows the user application to check the integrity of the UF and RF connection lines between the STCU2 and the FCCU. Refer to the description of FCCU fault injection mechanism to understand how the UF/RF set/clear mechanism works.

63.8.7 Watchdogs

The STCU2 implements three different watchdogs to ensure that operations are finished in time.

63.8.7.1 Initialization watchdog timer

The STCU2 has two initialization operating modes: program the self-test sequence and run it (Safety mode) or bypass completely the self-test sequence setting the bit STCU2_CFG.BYP. In case there are faults during the initialization phase preventing the selection of one of these two operating modes, a hard-coded watchdog time-out flags the wrong behavior.

63.8.7.2 Built-in self-test watchdog timer

The LBIST and MBIST execution time has to be configured as described in [STCU2 Watchdog Granularity \(STCU2_WDG\)](#), to account for the overall execution time of the Self-Test sequence. In case the selected LBISTs or MBISTs are not yet completed during the Off-line Self-Test assigned time, the current LBISTs or MBISTs execution is interrupted and a failure is flagged into STCU2_ERR_STAT.WDTO and STCU2_MBEL/M/H or STCU2_LBE registers while in case of On-Line Self-Test into STCU2_ERR_STAT.WDTOSW and STCU2_MBELSW/MSW/HSW or STCU2_LBESW.

In case of multiple sequential run in the same On-Line/Off-Line session and the time-out happens in the middle of a sequential run, the next sequential run will be skipped and the execution ends with the current updated status of the registers reported above.

In all the cases, the STCU2 reset generation is managed in one of the modes described in [Reset management](#).

63.8.7.3 Register write-access watchdog timer

As explained in the STCU2_SKC register description:

- A key mechanism protects STCU2 registers during the self-test configuration phase (both online/offline) by preventing any unwanted access.
- A hardware watchdog timer locks register-write access after a number of STCU2 clock cycles. To refresh the hardware watchdog timer before it times out, write only security key 2 to STCU2_SKC.

AUTOLOCK_VALUE is the number of STCU2 clock cycles after which the hardware watchdog timer locks register-write access. See the chip-specific STCU2 information for the value of AUTOLOCK_VALUE.

NOTE

For write completion, each DCF record requires a number of STCU2 clock cycles equal to DCF_COMPLETION. The maximum number of DCF records that can be executed before the watchdog timer must be refreshed is AUTOLOCK_VALUE divided by DCF_COMPLETION. See the chip-specific STCU2 information for the value of DCF_COMPLETION.

The hardware watchdog timer is particularly useful in case STCU2_CFG[WRP] is 0 during the software self-test configuration. In this case, the software application might enable write access to the STCU2 registers.

63.8.8 SSCM interface

The SSCM interface is used to program the STCU2's configuration parameters without the CPU intervention after a reset trigger event initializes the STCU2. This bus interface has the priority over the IPS.

63.9 Register description

During Offline SELFTEST, once the STCU2 registers are configured they cannot be overridden.

During Online SELFTEST, once the STCU2 registers are configured they cannot be overridden via IPS till the selftest is complete.

The STCU2 registers are listed in this section.

STCU2 memory map

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
0	STCU2 Run Register (STCU2_RUN)	32	R/W	0000_0000h	63.9.1/3166
4	STCU2 Run Software Register (STCU2_RUNSW)	32	R/W	0000_0000h	63.9.2/3167
8	STCU2 SK Code Register (STCU2_SKC)	32	W	0000_0000h	63.9.3/3169
C	STCU2 Configuration Register (STCU2_CFG)	32	R/W	See section	63.9.4/3170
10	STCU2 PLL Configuration Register (STCU2_PLL_CFG)	32	R/W	See section	63.9.5/3173
14	STCU2 Watchdog Granularity (STCU2_WDG)	32	R/W	See section	63.9.6/3174
18	STCU2 Interrupt Flag Register (STCU2_INT_FLG)	32	w1c	0000_0000h	63.9.7/3176
24	STCU2 Error Register (STCU2_ERR_STAT)	32	R/W	0000_0000h	63.9.8/3177
28	STCU2 Error FM Register (STCU2_ERR_FM)	32	R/W	See section	63.9.9/3180
2C	STCU2 Off-Line LBIST Status Register (STCU2_LBS)	32	R	0000_0000h	63.9.10/3181
30	STCU2 Off-Line LBIST End Flag Register (STCU2_LBE)	32	R	0000_0000h	63.9.11/3182
34	STCU2 On-Line LBIST Status Register (STCU2_LBSSW)	32	R	0000_0000h	63.9.12/3183
38	STCU2 On-Line LBIST End Flag Register (STCU2_LBESW)	32	R	0000_0000h	63.9.13/3185
3C	STCU2 On-Line LBIST Reset Management (STCU2_LBRMSW)	32	R/W	See section	63.9.14/3186
40	STCU2 LBIST Unrecoverable FM Register (STCU2_LBUFM)	32	R/W	See section	63.9.15/3188

Table continues on the next page...

STCU2 memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
44	STCU2 Off-Line MBIST Status Low Register (STCU2_MBSL)	32	R	0000_0000h	63.9.16/ 3189
48	STCU2 Off-Line MBIST Status Medium Register (STCU2_MBSM)	32	R	0000_0000h	63.9.17/ 3194
4C	STCU2 Off-Line MBIST Status High Register (STCU2_MBSH)	32	R	0000_0000h	63.9.18/ 3199
50	STCU2 Off-Line MBIST End Flag Low Register (STCU2_MBEL)	32	R	0000_0000h	63.9.19/ 3200
54	STCU2 Off-Line MBIST End Flag Medium Register (STCU2_MBEM)	32	R	0000_0000h	63.9.20/ 3204
58	STCU2 Off-Line MBIST End Flag High Register (STCU2_MBEH)	32	R	0000_0000h	63.9.21/ 3208
5C	STCU2 On-Line MBIST Status Low Register (STCU2_MBSLSW)	32	R	0000_0000h	63.9.22/ 3209
60	STCU2 On-Line MBIST Status Medium Register (STCU2_MBSMSW)	32	R	0000_0000h	63.9.23/ 3212
64	STCU2 On-Line MBIST Status High Register (STCU2_MBSHSW)	32	R	0000_0000h	63.9.24/ 3216
68	STCU2 On-Line MBIST End Flag Low Register (STCU2_MBELSW)	32	R	0000_0000h	63.9.25/ 3217
6C	STCU2 On-Line MBIST End Flag Medium Register (STCU2_MBEMSW)	32	R	0000_0000h	63.9.26/ 3221
70	STCU2 On-Line MBIST End Flag High Register (STCU2_MBEHSW)	32	R	0000_0000h	63.9.27/ 3225
74	STCU2 MBIST Unrecoverable FM Low Register (STCU2_MBUFML)	32	R/W	See section	63.9.28/ 3226
78	STCU2 MBIST Unrecoverable FM Medium Register (STCU2_MBUFMM)	32	R/W	See section	63.9.29/ 3229
7C	STCU2 MBIST Unrecoverable FM High Register (STCU2_MBUFMH)	32	R/W	See section	63.9.30/ 3233
100	STCU2 LBIST Control (STCU2_LB_CTRL0)	32	R/W	See section	63.9.31/ 3234
104	STCU2 LBIST PC Stop Register (STCU2_LB_PCS0)	32	R/W	See section	63.9.32/ 3237
110	STCU2 Off-Line LBIST MISR Expected Low Register (STCU2_LB_MISREL0)	32	R/W	See section	63.9.33/ 3238
114	STCU2 Off-Line LBIST MISR Expected High Register (STCU2_LB_MISREH0)	32	R/W	See section	63.9.34/ 3238
118	STCU2 Off-Line LBIST MISR Read Low Register (STCU2_LB_MISRRL0)	32	R	See section	63.9.35/ 3239
11C	STCU2 Off-Line LBIST MISR Read High Register (STCU2_LB_MISRRLH0)	32	R	See section	63.9.36/ 3240
120	STCU2 On-Line LBIST MISR Expected Low Register (STCU2_LB_MISRELSW0)	32	R/W	See section	63.9.37/ 3240

Table continues on the next page...

STCU2 memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
124	STCU2 On-Line LBIST MISR Expected High Register (STCU2_LB_MISREHSW0)	32	R/W	See section	63.9.38/ 3241
128	STCU2 On-Line LBIST MISR Read Low Register (STCU2_LB_MISRRLSW0)	32	R	See section	63.9.39/ 3242
12C	STCU2 On-Line LBIST MISR Read High Register (STCU2_LB_MISRRLHSW0)	32	R	See section	63.9.40/ 3242
140	STCU2 LBIST Control (STCU2_LB_CTRL1)	32	R/W	See section	63.9.31/ 3234
144	STCU2 LBIST PC Stop Register (STCU2_LB_PCS1)	32	R/W	See section	63.9.32/ 3237
150	STCU2 Off-Line LBIST MISR Expected Low Register (STCU2_LB_MISREL1)	32	R/W	See section	63.9.33/ 3238
154	STCU2 Off-Line LBIST MISR Expected High Register (STCU2_LB_MISREH1)	32	R/W	See section	63.9.34/ 3238
158	STCU2 Off-Line LBIST MISR Read Low Register (STCU2_LB_MISRRL1)	32	R	See section	63.9.35/ 3239
15C	STCU2 Off-Line LBIST MISR Read High Register (STCU2_LB_MISRRLH1)	32	R	See section	63.9.36/ 3240
160	STCU2 On-Line LBIST MISR Expected Low Register (STCU2_LB_MISRELSW1)	32	R/W	See section	63.9.37/ 3240
164	STCU2 On-Line LBIST MISR Expected High Register (STCU2_LB_MISREHSW1)	32	R/W	See section	63.9.38/ 3241
168	STCU2 On-Line LBIST MISR Read Low Register (STCU2_LB_MISRRLSW1)	32	R	See section	63.9.39/ 3242
16C	STCU2 On-Line LBIST MISR Read High Register (STCU2_LB_MISRRLHSW1)	32	R	See section	63.9.40/ 3242
180	STCU2 LBIST Control (STCU2_LB_CTRL2)	32	R/W	See section	63.9.31/ 3234
184	STCU2 LBIST PC Stop Register (STCU2_LB_PCS2)	32	R/W	See section	63.9.32/ 3237
190	STCU2 Off-Line LBIST MISR Expected Low Register (STCU2_LB_MISREL2)	32	R/W	See section	63.9.33/ 3238
194	STCU2 Off-Line LBIST MISR Expected High Register (STCU2_LB_MISREH2)	32	R/W	See section	63.9.34/ 3238
198	STCU2 Off-Line LBIST MISR Read Low Register (STCU2_LB_MISRRL2)	32	R	See section	63.9.35/ 3239
19C	STCU2 Off-Line LBIST MISR Read High Register (STCU2_LB_MISRRLH2)	32	R	See section	63.9.36/ 3240
1A0	STCU2 On-Line LBIST MISR Expected Low Register (STCU2_LB_MISRELSW2)	32	R/W	See section	63.9.37/ 3240
1A4	STCU2 On-Line LBIST MISR Expected High Register (STCU2_LB_MISREHSW2)	32	R/W	See section	63.9.38/ 3241
1A8	STCU2 On-Line LBIST MISR Read Low Register (STCU2_LB_MISRRLSW2)	32	R	See section	63.9.39/ 3242

Table continues on the next page...

STCU2 memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
1AC	STCU2 On-Line LBIST MISR Read High Register (STCU2_LB_MISRHSW2)	32	R	See section	63.9.40/3242
1C0	STCU2 LBIST Control (STCU2_LB_CTRL3)	32	R/W	See section	63.9.31/3234
1C4	STCU2 LBIST PC Stop Register (STCU2_LB_PCS3)	32	R/W	See section	63.9.32/3237
1D0	STCU2 Off-Line LBIST MISR Expected Low Register (STCU2_LB_MISREL3)	32	R/W	See section	63.9.33/3238
1D4	STCU2 Off-Line LBIST MISR Expected High Register (STCU2_LB_MISREH3)	32	R/W	See section	63.9.34/3238
1D8	STCU2 Off-Line LBIST MISR Read Low Register (STCU2_LB_MISRRL3)	32	R	See section	63.9.35/3239
1DC	STCU2 Off-Line LBIST MISR Read High Register (STCU2_LB_MISRHH3)	32	R	See section	63.9.36/3240
1E0	STCU2 On-Line LBIST MISR Expected Low Register (STCU2_LB_MISRELSW3)	32	R/W	See section	63.9.37/3240
1E4	STCU2 On-Line LBIST MISR Expected High Register (STCU2_LB_MISREHSW3)	32	R/W	See section	63.9.38/3241
1E8	STCU2 On-Line LBIST MISR Read Low Register (STCU2_LB_MISRRLSW3)	32	R	See section	63.9.39/3242
1EC	STCU2 On-Line LBIST MISR Read High Register (STCU2_LB_MISRHSW3)	32	R	See section	63.9.40/3242
200	STCU2 LBIST Control (STCU2_LB_CTRL4)	32	R/W	See section	63.9.31/3234
204	STCU2 LBIST PC Stop Register (STCU2_LB_PCS4)	32	R/W	See section	63.9.32/3237
210	STCU2 Off-Line LBIST MISR Expected Low Register (STCU2_LB_MISREL4)	32	R/W	See section	63.9.33/3238
214	STCU2 Off-Line LBIST MISR Expected High Register (STCU2_LB_MISREH4)	32	R/W	See section	63.9.34/3238
218	STCU2 Off-Line LBIST MISR Read Low Register (STCU2_LB_MISRRL4)	32	R	See section	63.9.35/3239
21C	STCU2 Off-Line LBIST MISR Read High Register (STCU2_LB_MISRHH4)	32	R	See section	63.9.36/3240
220	STCU2 On-Line LBIST MISR Expected Low Register (STCU2_LB_MISRELSW4)	32	R/W	See section	63.9.37/3240
224	STCU2 On-Line LBIST MISR Expected High Register (STCU2_LB_MISREHSW4)	32	R/W	See section	63.9.38/3241
228	STCU2 On-Line LBIST MISR Read Low Register (STCU2_LB_MISRRLSW4)	32	R	See section	63.9.39/3242
22C	STCU2 On-Line LBIST MISR Read High Register (STCU2_LB_MISRHSW4)	32	R	See section	63.9.40/3242
240	STCU2 LBIST Control (STCU2_LB_CTRL5)	32	R/W	See section	63.9.31/3234

Table continues on the next page...

STCU2 memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
244	STCU2 LBIST PC Stop Register (STCU2_LB_PCS5)	32	R/W	See section	63.9.32/ 3237
250	STCU2 Off-Line LBIST MISR Expected Low Register (STCU2_LB_MISREL5)	32	R/W	See section	63.9.33/ 3238
254	STCU2 Off-Line LBIST MISR Expected High Register (STCU2_LB_MISREH5)	32	R/W	See section	63.9.34/ 3238
258	STCU2 Off-Line LBIST MISR Read Low Register (STCU2_LB_MISRRL5)	32	R	See section	63.9.35/ 3239
25C	STCU2 Off-Line LBIST MISR Read High Register (STCU2_LB_MISRRH5)	32	R	See section	63.9.36/ 3240
260	STCU2 On-Line LBIST MISR Expected Low Register (STCU2_LB_MISRELSW5)	32	R/W	See section	63.9.37/ 3240
264	STCU2 On-Line LBIST MISR Expected High Register (STCU2_LB_MISREHSW5)	32	R/W	See section	63.9.38/ 3241
268	STCU2 On-Line LBIST MISR Read Low Register (STCU2_LB_MISRRLSW5)	32	R	See section	63.9.39/ 3242
26C	STCU2 On-Line LBIST MISR Read High Register (STCU2_LB_MISRRHSW5)	32	R	See section	63.9.40/ 3242
600	STCU2 MBIST Control Register (STCU2_MB_CTRL0)	32	R/W	See section	63.9.41/ 3243
604	STCU2 MBIST Control Register (STCU2_MB_CTRL1)	32	R/W	See section	63.9.41/ 3243
608	STCU2 MBIST Control Register (STCU2_MB_CTRL2)	32	R/W	See section	63.9.41/ 3243
60C	STCU2 MBIST Control Register (STCU2_MB_CTRL3)	32	R/W	See section	63.9.41/ 3243
610	STCU2 MBIST Control Register (STCU2_MB_CTRL4)	32	R/W	See section	63.9.41/ 3243
614	STCU2 MBIST Control Register (STCU2_MB_CTRL5)	32	R/W	See section	63.9.41/ 3243
618	STCU2 MBIST Control Register (STCU2_MB_CTRL6)	32	R/W	See section	63.9.41/ 3243
61C	STCU2 MBIST Control Register (STCU2_MB_CTRL7)	32	R/W	See section	63.9.41/ 3243
620	STCU2 MBIST Control Register (STCU2_MB_CTRL8)	32	R/W	See section	63.9.41/ 3243
624	STCU2 MBIST Control Register (STCU2_MB_CTRL9)	32	R/W	See section	63.9.41/ 3243
628	STCU2 MBIST Control Register (STCU2_MB_CTRL10)	32	R/W	See section	63.9.41/ 3243
62C	STCU2 MBIST Control Register (STCU2_MB_CTRL11)	32	R/W	See section	63.9.41/ 3243
630	STCU2 MBIST Control Register (STCU2_MB_CTRL12)	32	R/W	See section	63.9.41/ 3243

Table continues on the next page...

STCU2 memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
634	STCU2 MBIST Control Register (STCU2_MB_CTRL13)	32	R/W	See section	63.9.41/3243
638	STCU2 MBIST Control Register (STCU2_MB_CTRL14)	32	R/W	See section	63.9.41/3243
63C	STCU2 MBIST Control Register (STCU2_MB_CTRL15)	32	R/W	See section	63.9.41/3243
640	STCU2 MBIST Control Register (STCU2_MB_CTRL16)	32	R/W	See section	63.9.41/3243
644	STCU2 MBIST Control Register (STCU2_MB_CTRL17)	32	R/W	See section	63.9.41/3243
648	STCU2 MBIST Control Register (STCU2_MB_CTRL18)	32	R/W	See section	63.9.41/3243
64C	STCU2 MBIST Control Register (STCU2_MB_CTRL19)	32	R/W	See section	63.9.41/3243
650	STCU2 MBIST Control Register (STCU2_MB_CTRL20)	32	R/W	See section	63.9.41/3243
654	STCU2 MBIST Control Register (STCU2_MB_CTRL21)	32	R/W	See section	63.9.41/3243
658	STCU2 MBIST Control Register (STCU2_MB_CTRL22)	32	R/W	See section	63.9.41/3243
65C	STCU2 MBIST Control Register (STCU2_MB_CTRL23)	32	R/W	See section	63.9.41/3243
660	STCU2 MBIST Control Register (STCU2_MB_CTRL24)	32	R/W	See section	63.9.41/3243
664	STCU2 MBIST Control Register (STCU2_MB_CTRL25)	32	R/W	See section	63.9.41/3243
668	STCU2 MBIST Control Register (STCU2_MB_CTRL26)	32	R/W	See section	63.9.41/3243
66C	STCU2 MBIST Control Register (STCU2_MB_CTRL27)	32	R/W	See section	63.9.41/3243
670	STCU2 MBIST Control Register (STCU2_MB_CTRL28)	32	R/W	See section	63.9.41/3243
674	STCU2 MBIST Control Register (STCU2_MB_CTRL29)	32	R/W	See section	63.9.41/3243
678	STCU2 MBIST Control Register (STCU2_MB_CTRL30)	32	R/W	See section	63.9.41/3243
67C	STCU2 MBIST Control Register (STCU2_MB_CTRL31)	32	R/W	See section	63.9.41/3243
680	STCU2 MBIST Control Register (STCU2_MB_CTRL32)	32	R/W	See section	63.9.41/3243
684	STCU2 MBIST Control Register (STCU2_MB_CTRL33)	32	R/W	See section	63.9.41/3243
688	STCU2 MBIST Control Register (STCU2_MB_CTRL34)	32	R/W	See section	63.9.41/3243

Table continues on the next page...

STCU2 memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
68C	STCU2 MBIST Control Register (STCU2_MB_CTRL35)	32	R/W	See section	63.9.41/3243
690	STCU2 MBIST Control Register (STCU2_MB_CTRL36)	32	R/W	See section	63.9.41/3243
694	STCU2 MBIST Control Register (STCU2_MB_CTRL37)	32	R/W	See section	63.9.41/3243
698	STCU2 MBIST Control Register (STCU2_MB_CTRL38)	32	R/W	See section	63.9.41/3243
69C	STCU2 MBIST Control Register (STCU2_MB_CTRL39)	32	R/W	See section	63.9.41/3243
6A0	STCU2 MBIST Control Register (STCU2_MB_CTRL40)	32	R/W	See section	63.9.41/3243
6A4	STCU2 MBIST Control Register (STCU2_MB_CTRL41)	32	R/W	See section	63.9.41/3243
6A8	STCU2 MBIST Control Register (STCU2_MB_CTRL42)	32	R/W	See section	63.9.41/3243
6AC	STCU2 MBIST Control Register (STCU2_MB_CTRL43)	32	R/W	See section	63.9.41/3243
6B0	STCU2 MBIST Control Register (STCU2_MB_CTRL44)	32	R/W	See section	63.9.41/3243
6B4	STCU2 MBIST Control Register (STCU2_MB_CTRL45)	32	R/W	See section	63.9.41/3243
6B8	STCU2 MBIST Control Register (STCU2_MB_CTRL46)	32	R/W	See section	63.9.41/3243
6BC	STCU2 MBIST Control Register (STCU2_MB_CTRL47)	32	R/W	See section	63.9.41/3243
6C0	STCU2 MBIST Control Register (STCU2_MB_CTRL48)	32	R/W	See section	63.9.41/3243
6C4	STCU2 MBIST Control Register (STCU2_MB_CTRL49)	32	R/W	See section	63.9.41/3243
6C8	STCU2 MBIST Control Register (STCU2_MB_CTRL50)	32	R/W	See section	63.9.41/3243
6CC	STCU2 MBIST Control Register (STCU2_MB_CTRL51)	32	R/W	See section	63.9.41/3243
6D0	STCU2 MBIST Control Register (STCU2_MB_CTRL52)	32	R/W	See section	63.9.41/3243
6D4	STCU2 MBIST Control Register (STCU2_MB_CTRL53)	32	R/W	See section	63.9.41/3243
6D8	STCU2 MBIST Control Register (STCU2_MB_CTRL54)	32	R/W	See section	63.9.41/3243
6DC	STCU2 MBIST Control Register (STCU2_MB_CTRL55)	32	R/W	See section	63.9.41/3243
6E0	STCU2 MBIST Control Register (STCU2_MB_CTRL56)	32	R/W	See section	63.9.41/3243

Table continues on the next page...

STCU2 memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
6E4	STCU2 MBIST Control Register (STCU2_MB_CTRL57)	32	R/W	See section	63.9.41/3243
6E8	STCU2 MBIST Control Register (STCU2_MB_CTRL58)	32	R/W	See section	63.9.41/3243
6EC	STCU2 MBIST Control Register (STCU2_MB_CTRL59)	32	R/W	See section	63.9.41/3243
6F0	STCU2 MBIST Control Register (STCU2_MB_CTRL60)	32	R/W	See section	63.9.41/3243
6F4	STCU2 MBIST Control Register (STCU2_MB_CTRL61)	32	R/W	See section	63.9.41/3243
6F8	STCU2 MBIST Control Register (STCU2_MB_CTRL62)	32	R/W	See section	63.9.41/3243
6FC	STCU2 MBIST Control Register (STCU2_MB_CTRL63)	32	R/W	See section	63.9.41/3243
700	STCU2 MBIST Control Register (STCU2_MB_CTRL64)	32	R/W	See section	63.9.41/3243

63.9.1 STCU2 Run Register (STCU2_RUN)

The STCU2_RUN register defines the RUN bit to start the off-line self testing procedure.

The R/W fields in this register are readable at any time. However, you can write to these fields only when Off-line Self-Test phase is still active.

Address: 0h base + 0h offset = 0h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0								0							
W						BYP	MBPLEN	LBPLEN								
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

STCU2_RUN field descriptions

Field	Description
0–20 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
21 BYP	Bypass mode 0 Off-Line Self-Test is active. In case the RUN bit into the STCU2_RUN register is not set before the Hard-Coded WDG Time-out, the WDTO error is generated into the STCU2_ERR_STAT register. 1 Off-Line Self-Test is completely bypassed and the access to STCU2 is locked until the STCU2_SKC register is written according to the request access mode.
22 MBPLEN	Off-Line MBIST with PLL Enabled 0 Off-Line MBIST is executed without using the on-chip PLL. 1 Off-Line MBIST is executed enabling the on-chip PLL control interface selecting the parameters defined into STCU2_PLL_CFG register.
23 LBPLEN	Off-Line LBIST with PLL Enabled 0 Off-Line LBIST is executed without using the on-chip PLL. 1 Off-Line LBIST is executed enabling the on-chip PLL control interface selecting the parameters defined into STCU2_PLL_CFG register.
24–30 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
31 RUN	RUN: RUN The RUN bit is automatically cleared by the STCU2 when the off-line self testing procedure has been completed. 0 Idle 1 Off-Line Self testing procedure is running

63.9.2 STCU2 Run Software Register (STCU2_RUNSW)

The STCU2_RUNSW register defines the RUN bit to start the on-line self testing procedure.

The R/W fields in this register are readable at any time. However, you can write to these fields only when both of the following conditions are true:

- STCU2_CFG[WRP] = 0
- On-line Self-Test phase is active. The On-Line condition starts when the Off-Line is over. In this condition, system is alive and the SW can program the STCU2

Register description

Address: 0h base + 4h offset = 4h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31		
R	0					0	MBSWPLEN	LBSWPLEN	0								0	RUNSW
W					MBIE													
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0		

STCU2_RUNSW field descriptions

Field	Description
0–19 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
20 MBIE	MBIST Interrupt Enable 0 Interrupt is not generated at the end of the software MBIST execution phase of the selected Memories. 1 At the end of the software MBIST execution phase of the concurrent selected memories the STCU2 forces an interrupt request to notify that MBISTs of concurrent selected memories are over. In case there is a sequence of concurrent runs in the same Self-Test run, the interrupt request is generated at the end of each concurrent runs provided that the software will clear the STCU2_INT_FLG[MBIFG] bit in between.
21 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
22 MBSWPLEN	On-Line MBIST with PLL Enabled 0 On-Line MBIST is executed without using the on-chip PLL. 1 On-Line MBIST is executed using the PLL configuration provided by software. STCU2 does not take the PLL control but monitors the PLL lock signal to check if PLL is working correctly.
23 LBSWPLEN	On-Line LBIST with PLL Enabled 0 On-Line LBIST is executed without using the on-chip PLL. 1 On-Line LBIST is executed using the PLL configuration provided by software. STCU2 does not take the PLL control but monitors the PLL lock signal to check if PLL is working correctly.
24–29 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
30 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
31 RUNSW	RUNSW The RUNSW bit is automatically cleared by the STCU2 when the on-line self testing procedure has been completed. 0 Idle 1 On-line self testing procedure is running

63.9.3 STCU2 SK Code Register (STCU2_SKC)

The STCU2_SKC register implements the security key code mechanism needed to access in write mode to the other STCU2 registers. In order to unlock the STCU2 access after:

- The STCU2 asynchronous reset
- The end of the STCU2 run

the software (IPS bus) or the SSCM interfaces have to apply the following sequence:

- write the key1 into the STCU2_SKC register
- write the key2 into the STCU2_SKC register

Depending on the off/on-line test step, the two keys will be different. Byte write operation is not allowed since the full key has to be recognized as one unit.

It should be noted that after the Self-Test sequence has been completed or the Bypass feature has been enabled (setting the bit BYP into the STCU2_CFG), the SSCM interface is no longer available.

In case of invalid access or sequence (Key1/2 have to be applied consecutively), a transfer error on the IPS or SSCM bus is asserted depending on the selected source. The STCU2 write access is locked and to unlock the access the sequence has to be applied again.

In case the STCU2 register access lasts more cycles than the one defined into the Hard-coded WDG time-out, the STCU2 write access is locked and the WDG and Register ITF clocks are switched off. Also in this case, in order to enable again the write access to the STCU2 and the WDG and Register ITF clocks, it is required to apply again the sequence.

In case it is required to extend the STCU2 register access cycles before the Hard-coded WDG time-out expires, only the Key2 has to be applied. The effect of this write operation is to re-initialize the WDG time-out counter. In such a condition, Key1 has not to be applied otherwise a transfer error on the IPS bus is asserted depending on the selected source. The STCU2 write access is locked and to unlock the access the sequence has to be applied again.

The STCU2_SKC register is not readable. The value 00000000h is always returned in case of read operation.

Register description

Address: 0h base + 8h offset = 8h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R																																
W																SKC																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	

STCU2_SKC field descriptions

Field	Description
0–31 SKC	STCU2 security key code for off-line Test = D3FEA98Bh: Key1 to unlock the write access the STCU2 = 2C015674h: Key2 to unlock the write access the STCU2 STCU2 security key code for on-line Test = 753F924Eh: Key1 to unlock the write access the STCU2 = 8AC06DB1h: Key2 to unlock the write access the STCU2

63.9.4 STCU2 Configuration Register (STCU2_CFG)

NOTE

The reset value is chip-specific; see the chapter that describes how modules are configured and connected.

The STCU2_CFG register includes the global configuration of the STCU2 and can be updated both in the Off/On-Line Test steps.

The access to this register is described in the following figure. It further depends on the state of the WRP bit as follows:

- When WRP = 0: The register can be written during the on-line Self-Test case without restrictions
- When WRP = 1:
 - The only bit that can be written without any restriction is WRP.
 - In case there are software operations that write all the register's bits, a transfer error is raised only if the value of the selected byte written differs from the current status of the register. This functionality has been implemented to prevent potential compilation behavior that might invalidate this single bit clean capability.

MBIST can be run in three different test types according to how the STCU2_CFG register bits 27 and 28 are set.

The following table describes the test algorithm and coverage of the different test types.

Table 63-10. MBIST algorithm and coverage according to test type

MBIST Type	STCU2_CFG[PM OSEN]	STCU2_CFG[MB U]	MBIST Algorithm	Internal Nodes/ Circuits Additionally Covered by Test	Use Case	Test Cycle
Full Test	1	0	Test memory using all built in algorithms. Open PMOS algorithm included	Address Decoder and Bitcell as well as resistive defects in the address decoder	Used by MCU production test. Recommended to use in the field for MBIST online self-test.	About 3 to 4x more than Auto Test
Reduced Test	0	0	Test memory using all built in algorithms except the open PMOS algorithm.	Address Decoder and Bitcell	Recommended to use in the field for MBIST online self-test if the Full Test takes too long.	About 10% less than Full Test
Auto Test	x	1	A smaller set of algorithms which target latent defect mechanisms.	Latent defects such as NBTI of the PMOS transistors in the bitcells	Recommended to use for MBIST offline self-test as an optimum balance of test time vs. fault coverage.	About 3 to 4x faster than Full Test

Address: 0h base + Ch offset = Ch

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	PTR							LB_DELAY							
W																
Reset	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*
Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0							WRP	0	0	PMOSEN	MBU	CLK_CFG			
W																
Reset	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*

* Notes:

- The reset value is chip-specific; see the chapter that describes how modules are configured and connected. x = Undefined at reset.

STCU2_CFG field descriptions

Field	Description
0 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
1–7 PTR	LBIST or MBIST pointer PTR defines the logical pointer to the first LBIST or MBIST to be scheduled when the self testing procedure is enabled. PTR is the entry pointer to a linked list of BIST descriptors. If PTR = 0 then the LBIST0 is initially scheduled.

Table continues on the next page...

STCU2_CFG field descriptions (continued)

Field	Description
	You can always read this field. In the off-line Self-Test phase, you can always write to this field. In the on-line Self-Test phase, you can write to this field only when WRP = 0. 0h to (LBIST -1): pointer to LBIST 10h to (10h + MBIST -1): pointer to MBIST 7Fh: pointer to NIL. No BIST execution.
8–15 LB_DELAY	Delay LBIST run LB_DELAY defines the delay between the LBIST starts when more than a single LBIST is selected to be executed concurrently with the purpose of smoothing the power consumption transient. The allowed delay time are the following: 00h: No delay 01h: 1 x16 STCU2 Core clock cycles 02h: 2 x 16 STCU2 Core clock cycles 03h: 3 x 16 STCU2 Core clock cycles ... FDh: 253 x 16 STCU2 Core clock cycles FEh: 254 x 16 STCU2 Core clock cycles FFh: 255 x 16 STCU2 Core clock cycles Note that in case the Frequency of the STCU2 core clock is 16 MHz, the delay parameters is expressed in μs. NOTE: This field should be written with 0xFF for self test run. You can always read this field. In the off-line Self-Test phase, you can always write to this field. In the on-line Self-Test phase, you can write to this field only when WRP = 0.
16–22 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
23 WRP	Write Protection 0: Specific STCU2 registers can be written through IPS bus interface after Off-Line Self-Test sequence has been completed 1: Specific STCU2 registers cannot be written though IPS also after Off-Line Self-Test sequence has been completed preventing in such a way any user application write operation
24–25 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
26 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
27 PMOSEN	MBIST PMOS Test Enable When MBU is 1 then "Auto test" MBIST test type will be selected regardless of the setting of this field. 0 When MBU is 0 then selects the "Reduced test" MBIST test type. 1 When MBU is 0 then selects the "Full test" MBIST test type.
28 MBU	MBIST MBU Test Enabled 0 Either "Full test" or "Reduced test" MBIST test type will be selected according to the PMOSEN value. 1 Selects "Auto test" MBIST test type regardless of the PMOSEN value.

Table continues on the next page...

STCU2_CFG field descriptions (continued)

Field	Description
29–31 CLK_CFG	Logic, Memory BIST and STCU2 core CLK Clock configuration CLK_CFG defines the ratio between the sys_clk and the TCK used to program both the LBIST and the MBIST and the STCU2 core clock. The punch-out mechanism is used to generate the derived clocks. The allowed configurations are the following: 000: sys_clk 001: sys_clk/2 010: sys_clk/3 ... 101: sys_clk/6 110: sys_clk/7 111: sys_clk/8 You can always read this field. In the off-line Self-Test phase, you can always write to this field. In the on-line Self-Test phase, you can write to this field only when WRP = 0.

63.9.5 STCU2 PLL Configuration Register (STCU2_PLL_CFG)**NOTE**

The reset value is chip-specific; see the chapter that describes how modules are configured and connected.

The STCU2_PLL_CFG register defines the parameters used to program the PLL only when the Off-line L/MBIST are performed and STCU2_RUN[MBPLEN] = 1 or STCU2_RUN[LBPLEN] = 1.

In the On-line condition, these bits are not effective. It is also possible to set the PLL during active On-line mode to prevent any potential functionality issue.

The R/W fields in this register are readable at any time. However, you can write to these fields only when Off-line Self-Test phase is still active.

Address: 0h base + 10h offset = 10h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	
R	0		PLLODF						0						PLLIDF			0										PLLIDF					
W																																	
Reset	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*		

* Notes:

- The reset value is chip-specific; see the chapter that describes how modules are configured and connected. x = Undefined at reset.

STCU2_PLL_CFG field descriptions

Field	Description
0–1 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
2–7 PLLODF	PLL Output Division Factor The value of this field drives the Output Division factor of the PLL embedded into the device. Refer to the PLL specifications to define the exact mapping between the PLLODF value and the related PLL's Output Division factor.
8–12 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
13–15 PLLIDF	PLL Input Division Factor The value of this field drives the Input Division factor of the PLL embedded into the device. Refer to the PLL specifications to define the exact mapping between the PLLIDF value and the related PLL's Input Division factor.
16–24 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
25–31 PLLLDF	PLL Loop Division Factor The value of this field drives the Loop Division factor of the PLL embedded into the device. Refer to the PLL specifications to define the exact mapping between the PLLLDF value and the related PLL's Loop Division factor.

63.9.6 STCU2 Watchdog Granularity (STCU2_WDG)**NOTE**

The reset value is chip-specific; see the chapter that describes how modules are configured and connected.

The STCU2_WDG register defines the time limit for LBIST and MBIST execution, providing a protection mechanism in case of deadlock or endless conditions during the self test procedure.

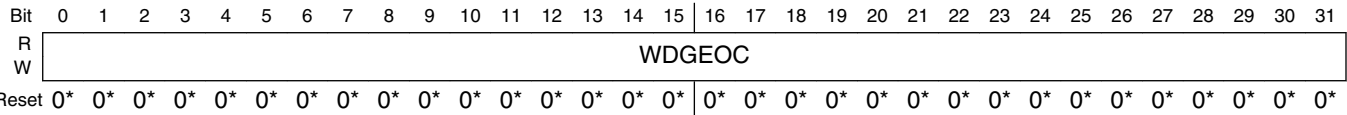
When the offline self test sequence is not run and the STCU2_CFG[BYP] field is not set, bits 15..0 define the time-out before the STCU2_ERR[WDTO] field is set and the STCU2 core clock is switched off.

In case offline/online self test sequence is run, the STCU2_WDG value defines the time-out of the execution run.

The register fields are always readable. You can write to these fields when either of the following conditions is true:

- Offline self test phase is active
- Online self test phase is active and CFG[WRP] = 0

Address: 0h base + 14h offset = 14h



- * Notes:
- The reset value is chip-specific; see the chapter that describes how modules are configured and connected.

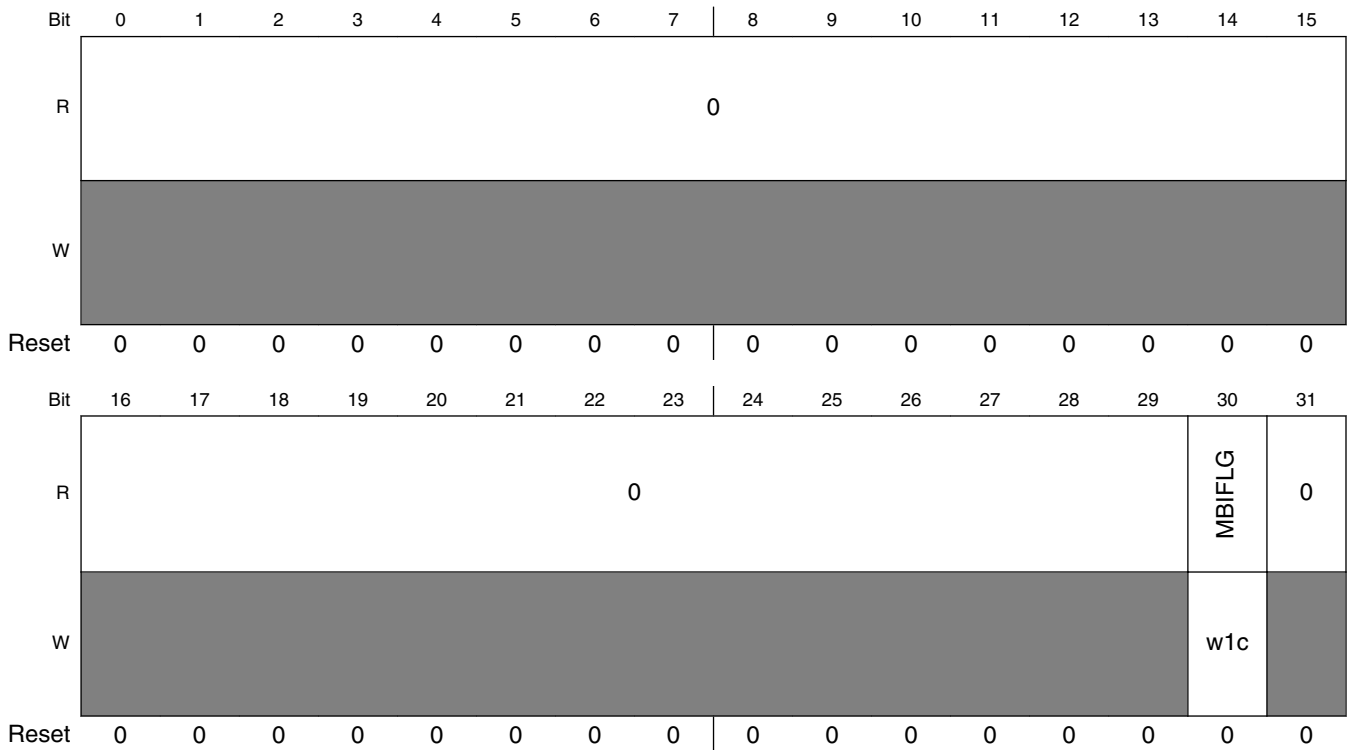
STCU2_WDG field descriptions

Field	Description
0–31 WDGEOC	Watchdog End of Count Timer This value defines the time limit for offline and online self test execution to ensure correct operation within the allowed time. NOTE: If this register is programmed while STCU2_RUN[RUN]=1, the new STCU2_WDG value, programmed after STCU2_RUN, does not take effect. The allowed values are: 0000 0000h: 1 × 16 STCU2 Core clock cycles 0000 0001h: 2 × 16 STCU2 Core clock cycles 0000 0002h: 3 × 16 STCU2 Core clock cycles ... FFFF FFFDh: 4294967294 × 16 STCU2 Core clock cycles FFFF FFFEh: 4294967295 × 16 STCU2 Core clock cycles FFFF FFFFh: 4294967296 × 16 STCU2 Core clock cycles

63.9.7 STCU2 Interrupt Flag Register (STCU2_INT_FLG)

The STCU2_INT_FLG register includes an MBIST interrupt pending bit. The bit is effective only during the On-Line Self-Test phase and is managed only when the related control bit in the STCU2_RUNSW register (MBIE) is enabled.

Address: 0h base + 18h offset = 18h



STCU2_INT_FLG field descriptions

Field	Description
0–29 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
30 MBIFLG	MBIST Interrupt Flag 0 No interrupt is pending 1 An interrupt, highlighting that the On-Line execution of the concurrent scheduled MBIST is completed, is pending. Write 1 to clear the bit.
31 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

63.9.8 STCU2 Error Register (STCU2_ERR_STAT)

The STCU2_ERR_STAT register includes the status flags related to the STCU2 internal fault conditions occurred during the configuration or the On/Off-Line self testing execution.

The UFSF and RFSF can be set/cleared using the FCCU dedicated channels.

The access to this register is described in the following figure and as follows:

- If you select the byte write capability to write only the UFSF and RFSF, then there is no restriction in writing these bits.
- If your software performs the write operations on other bits besides UFSF/RFSF, then a transfer error is generated only if the value you are writing to the other bits differs from their value currently stored in the register. This functionality has been implemented to prevent potential compilation behavior that might invalidate the UFSF/RFSF single bit set/clean capability.

Address: 0h base + 24h offset = 24h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0						0	0		0		LOCKESW	WDTOSW	0	ENGESW	INVPSW
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Register description

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0						UFSF	RFSF	0			LOCKE	WDTO	0	ENGE	INVP
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

STCU2_ERR_STAT field descriptions

Field	Description
0–5 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
6 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
7 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
8–10 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
11 LOCKESW	On-Line LOCK Error You can always read this field. The content of this field is initialized to its reset value as soon as STCU2_RUNSW[RUNSW] = 1. 0 In case PLL is enabled, it is correctly locked during the Self-Test sequence 1 When the PLL is enabled, this flag highlight that there has been an unexpected PLL unlock event during the On-Line Self-Test sequence execution. The On-Line Self-test run is stopped and the status of the current running LBISTs or MBISTs is saved into the related registers. The LOCK signal is monitored during the LBIST run when STCU2_RUNSW.LBSWPLEN is set and/or during the MBIST run when STCU2_RUNSW[MBSWPLEN] = 1.
12 WDTOSW	On-Line Watchdog time-out You can always read this field. The content of this field is initialized to its reset value as soon as STCU2_RUNSW[RUNSW] = 1. 0 LBIST and MBIST time slot have been completed within the assigned watchdog time. 1 LBIST and MBIST time slot haven't been completed within the assigned watchdog time or there are internal mismatches among End of Execution signals. The conditions that flag the failures are the following: • MBIST BEND status flags that at least one of the selected MBIST run is not finished • lbist_done of the selected LBIST flags the LBIST run is/are not finished
13 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
14 ENGESW	On-Line Engine Error

Table continues on the next page...

STCU2_ERR_STAT field descriptions (continued)

Field	Description
	You can always read this field. The content of this field is initialized to its reset value as soon as STCU2_RUNSW[RUNSW] = 1. 0 Valid Engine execution 1 Invalid Engine execution. The following conditions set this bit: • STCU2 state machines (Watchdog, Master and Loader/Shifter) selects an unused code • Serial pass/fail status of the selected MBIST and the global BBAD flag are not aligned • Serial bend status of the selected MBIST and the global BEND flag are not aligned
15 INVPSW	On-Line Invalid Pointer You can always read this field. The content of this field is initialized to its reset value as soon as STCU2_RUNSW[RUNSW] = 1. 0 Valid linked pointer list 1 Invalid linked pointer list. The following conditions set this bit: • Initial LBIST or MBIST pointer is out of range • LBIST is selected when MBIST is concurrently running or vice versa • Error in the LBIST/MBIST linking (execution generates an infinite loop)
16–21 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
22 UFSF	Unrecoverable Faults Status Flag This flag reports the global status of the UF. This field can be set or cleared using the FCCU dedicated channel, and can also be set or cleared by software. 0 No errors that trigger the Unrecoverable Faults condition 1 There are errors that trigger the Unrecoverable Faults condition
23 RFSF	Recoverable Faults Status Flag This flag reports the global status of the RF. This field can be set or cleared using the FCCU dedicated channel, and can also be set or cleared by software. 0 No errors that trigger the Recoverable Faults condition 1 There are errors that trigger the Recoverable Faults condition
24–26 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
27 LOCKE	Off-Line LOCK Error 0 The PLL is correctly locked during the Self-Test sequence 1 When the PLL is enabled, this flag highlights that there has been an unexpected PLL unlock event during the Off-Line Self-Test sequence execution. The Off-Line Self-test run is stopped and the status of the current running LBISTs or MBISTs is saved into the related registers. The LOCK signal is monitored during the LBIST run when STCU2_RUN[LBPLEN] = 1 and/or during the MBIST run when STCU2_RUN[MBPLEN] = 1.
28 WDTO	Off-Line Watchdog time-out 0 LBIST and MBIST time slot have been completed within the assigned watchdog time. 1 LBIST and MBIST time slot haven't been completed within the assigned watchdog time or there are internal mismatches among End of Execution signals. The conditions that flag the failures are the following: • The STCU2_RUN[RUN] bit or the STCU2_RUN[BYP] bit is not set before the Watchdog reaches the End of Count

Table continues on the next page...

STCU2_ERR_STAT field descriptions (continued)

Field	Description
	• MBIST BEND status flags that at least one of the selected MBIST run is not finished • lbist_done of the selected LBIST flags the LBIST run is/are not finished
29 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
30 ENGE	Off-Line Engine Error 0 Valid Engine execution 1 Invalid Engine execution. The following conditions set this bit: • STCU2 state machines (Watchdog, Master and Loader/Shifter) selects an unused code • Serial pass/fail status of the selected MBIST and the global BBAD flag are not aligned • Serial bend status of the selected MBIST and the global BEND flag are not aligned
31 INVP	Off-Line Invalid pointer 0 Valid linked pointer list 1 Invalid linked pointer list. The following conditions set this bit: • Initial LBIST or MBIST pointer is out of range • LBIST is selected when MBIST is concurrently running or vice versa • Error in the LBIST/MBIST linking (execution generates an infinite loop)

63.9.9 STCU2 Error FM Register (STCU2_ERR_FM)**NOTE**

The reset value is chip-specific; see the chapter that describes how modules are configured and connected.

The STCU2_ERR_FM register defines the fault mapping of the STCU2 faults described into the register STCU2_ERR_STAT in terms of Unrecoverable or Recoverable Fault. All sources of internal faults can be routed to UF and RF.

The R/W fields in this register are readable at any time. You can write to these fields when either of the following conditions is true:

- Off-line Self-Test phase is active
- On-line Self-Test phase is active and STCU2_CFG[WRP] = 0

Address: 0h base + 28h offset = 28h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0															
W																
Reset	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0											LOCKEUFM	WDTOUFM	0	ENGEUFM	INVPUFM
W																
Reset	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*

* Notes:

- The reset value is chip-specific; see the chapter that describes how modules are configured and connected.x = Undefined at reset.

STCU2_ERR_FM field descriptions

Field	Description
0–26 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
27 LOCKEUFM	PLL LOCK Unrecoverable Fault Mapping 0 Recoverable Fault Mapping 1 Unrecoverable Fault Mapping
28 WDTOUFM	Watchdog time-out Unrecoverable Fault Mapping 0 Recoverable Fault Mapping 1 Unrecoverable Fault Mapping
29 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
30 ENGEUFM	Engine Error Unrecoverable fault Mapping 0 Recoverable Fault Mapping 1 Unrecoverable Fault Mapping
31 INVPUFM	Invalid Pointer Unrecoverable Fault Mapping 0 Recoverable Fault Mapping 1 Unrecoverable Mapping

63.9.10 STCU2 Off-Line LBIST Status Register (STCU2_LBS)

The STCU2_LBS register includes the results corresponding to the execution of the selected Off-Line LBIST.

The size of the register depends on the number of LBIST .

Address: 0h base + 2Ch offset = 2Ch

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Register description

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0										LBS5	LBS4	LBS3	LBS2	LBS1	LBS0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

STCU2_LBS field descriptions

Field	Description
0–25 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
26 LBS5	Off-Line status of the selected LBIST 0 Failed LBIST execution 1 Successful LBIST execution
27 LBS4	Off-Line status of the selected LBIST 0 Failed LBIST execution 1 Successful LBIST execution
28 LBS3	Off-Line status of the selected LBIST 0 Failed LBIST execution 1 Successful LBIST execution
29 LBS2	Off-Line status of the selected LBIST 0 Failed LBIST execution 1 Successful LBIST execution
30 LBS1	Off-Line status of the selected LBIST 0 Failed LBIST execution 1 Successful LBIST execution
31 LBS0	Off-Line status of the selected LBIST 0 Failed LBIST execution 1 Successful LBIST execution

63.9.11 STCU2 Off-Line LBIST End Flag Register (STCU2_LBE)

The STCU2_LBE register includes the End Flag related to the execution of the selected Off-Line LBIST.

The size of the register depends on the number of LBIST .

Address: 0h base + 30h offset = 30h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0										LBE5	LBE4	LBE3	LBE2	LBE1	LBE0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

STCU2_LBE field descriptions

Field	Description
0–25 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
26 LBE5	Off-Line End status of the selected LBIST 0 LBIST execution not yet completed 1 LBIST execution finished
27 LBE4	Off-Line End status of the selected LBIST 0 LBIST execution not yet completed 1 LBIST execution finished
28 LBE3	Off-Line End status of the selected LBIST 0 LBIST execution not yet completed 1 LBIST execution finished
29 LBE2	Off-Line End status of the selected LBIST 0 LBIST execution not yet completed 1 LBIST execution finished
30 LBE1	Off-Line End status of the selected LBIST 0 LBIST execution not yet completed 1 LBIST execution finished
31 LBE0	Off-Line End status of the selected LBIST 0 LBIST execution not yet completed 1 LBIST execution finished

63.9.12 STCU2 On-Line LBIST Status Register (STCU2_LBSSW)

The STCU2_LBSSW register includes the results corresponding to the execution of the selected On-Line LBIST.

The size of the register depends on the number of LBIST .

The content of this register is initialized to its reset value as soon as STCU2_RUNSW[RUNSW] = 1.

Register description

Address: 0h base + 34h offset = 34h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0										LBSSW5	LBSSW4	LBSSW3	LBSSW2	LBSSW1	LBSSW0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

STCU2_LBSSW field descriptions

Field	Description
0–25 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
26 LBSSW5	LBSSWx: On-Line status of the selected LBIST 0 Failed LBIST execution 1 Successful LBIST execution
27 LBSSW4	LBSSWx: On-Line status of the selected LBIST 0 Failed LBIST execution 1 Successful LBIST execution
28 LBSSW3	LBSSWx: On-Line status of the selected LBIST 0 Failed LBIST execution 1 Successful LBIST execution
29 LBSSW2	LBSSWx: On-Line status of the selected LBIST 0 Failed LBIST execution 1 Successful LBIST execution
30 LBSSW1	LBSSWx: On-Line status of the selected LBIST

Table continues on the next page...

STCU2_LBSSW field descriptions (continued)

Field	Description
	0 Failed LBIST execution 1 Successful LBIST execution
31 LBSSW0	LBSSWx: On-Line status of the selected LBIST 0 Failed LBIST execution 1 Successful LBIST execution

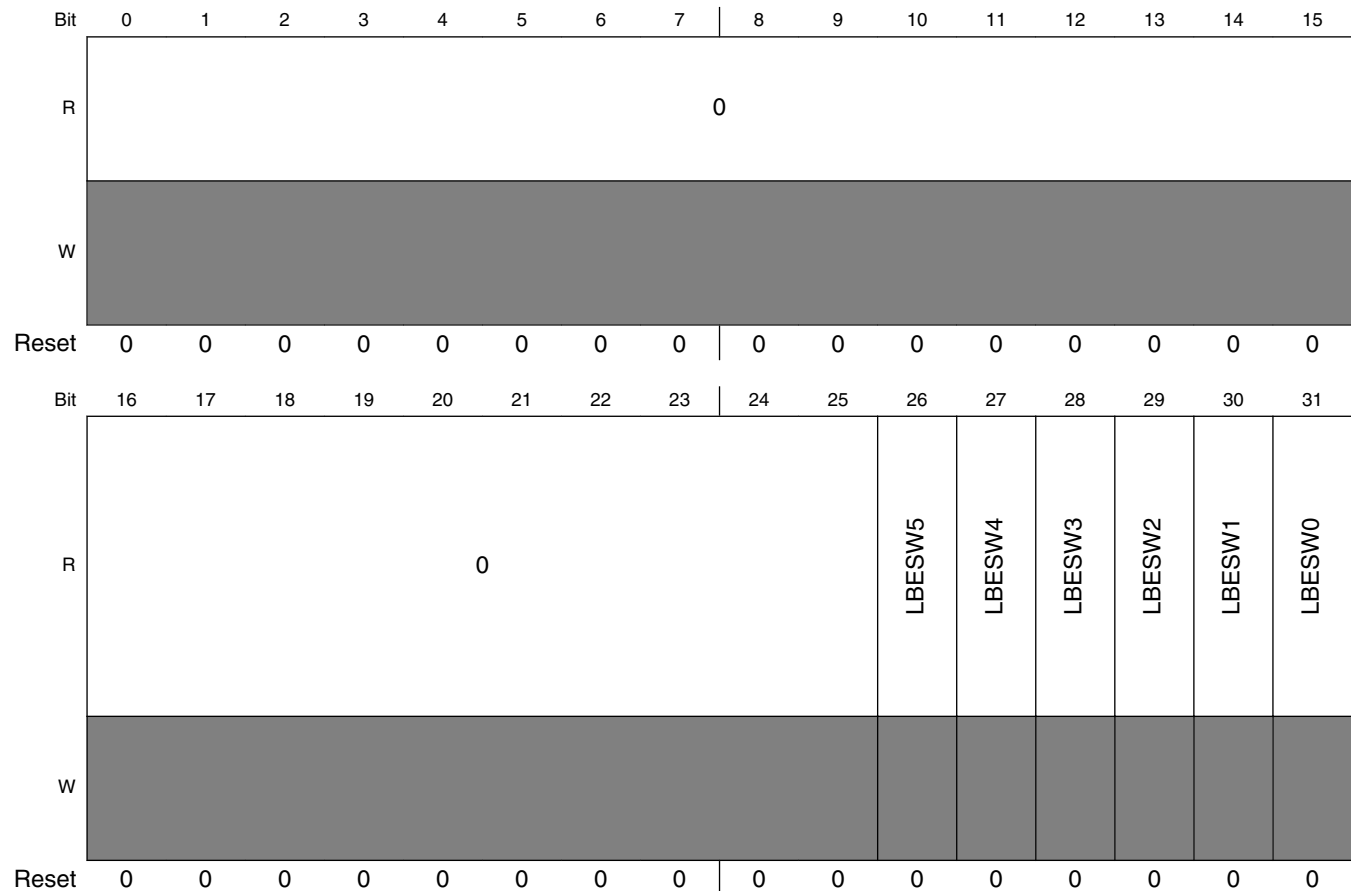
63.9.13 STCU2 On-Line LBIST End Flag Register (STCU2_LBESW)

The STCU2_LBESW register includes the End Flag related to the execution of the selected On-Line LBIST.

The size of the register depends on the number of LBIST.

The content of this register is initialized to its reset value as soon as STCU2_RUNSW[RUNSW] = 1.

Address: 0h base + 38h offset = 38h



STCU2_LBESW field descriptions

Field	Description
0–25 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
26 LBESW5	LBESWx: On-Line LBIST End status 0 LBIST execution not yet completed 1 LBIST execution finished
27 LBESW4	LBESWx: On-Line LBIST End status 0 LBIST execution not yet completed 1 LBIST execution finished
28 LBESW3	LBESWx: On-Line LBIST End status 0 LBIST execution not yet completed 1 LBIST execution finished
29 LBESW2	LBESWx: On-Line LBIST End status 0 LBIST execution not yet completed 1 LBIST execution finished
30 LBESW1	LBESWx: On-Line LBIST End status 0 LBIST execution not yet completed 1 LBIST execution finished
31 LBESW0	LBESWx: On-Line LBIST End status 0 LBIST execution not yet completed 1 LBIST execution finished

63.9.14 STCU2 On-Line LBIST Reset Management (STCU2_LBRMSW)**NOTE**

The reset value is chip-specific; see the chapter that describes how modules are configured and connected.

The STCU2_LBRMSW register defines the On-Line reset mapping for each LBIST. Global or specific functional reset can be programmed depending on the On-Line Self-Test requirements and top level integration architecture.

The size of the register depends on the number of LBIST .

The R/W fields in this register are readable at any time. You can write to these fields when either of the following conditions is true:

- Off-line Self-Test phase is active
- On-line Self-Test phase is active and STCU2_CFG[WRP] = 0

Address: 0h base + 3Ch offset = 3Ch

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0															
W																
Reset	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0										LBRMSW5	LBRMSW4	LBRMSW3	LBRMSW2	LBRMSW1	LBRMSW0
W																
Reset	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*

* Notes:

- The reset value is chip-specific; see the chapter that describes how modules are configured and connected. x = Undefined at reset.

STCU2_LBRMSW field descriptions

Field	Description
0–25 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
26 LBRMSW5	On-Line LBIST Reset Management NOTE: In case one of the selected LBIST has this bit set to '1', then only the Global functional reset will be pulsed. 0 Dedicated functional reset is pulsed at the end of LBIST run 1 Global functional reset is pulsed at the end of LBIST run
27 LBRMSW4	On-Line LBIST Reset Management NOTE: In case one of the selected LBIST has this bit set to '1', then only the Global functional reset will be pulsed. 0 Dedicated functional reset is pulsed at the end of LBIST run 1 Global functional reset is pulsed at the end of LBIST run
28 LBRMSW3	On-Line LBIST Reset Management NOTE: In case one of the selected LBIST has this bit set to '1', then only the Global functional reset will be pulsed. 0 Dedicated functional reset is pulsed at the end of LBIST run 1 Global functional reset is pulsed at the end of LBIST run
29 LBRMSW2	On-Line LBIST Reset Management NOTE: In case one of the selected LBIST has this bit set to '1', then only the Global functional reset will be pulsed. 0 Dedicated functional reset is pulsed at the end of LBIST run 1 Global functional reset is pulsed at the end of LBIST run

Table continues on the next page...

STCU2_LBRMSW field descriptions (continued)

Field	Description
30 LBRMSW1	On-Line LBIST Reset Management NOTE: In case one of the selected LBIST has this bit set to '1', then only the Global functional reset will be pulsed. 0 Dedicated functional reset is pulsed at the end of LBIST run 1 Global functional reset is pulsed at the end of LBIST run
31 LBRMSW0	On-Line LBIST Reset Management NOTE: In case one of the selected LBIST has this bit set to '1', then only the Global functional reset will be pulsed. 0 Dedicated functional reset is pulsed at the end of LBIST run 1 Global functional reset is pulsed at the end of LBIST run

63.9.15 STCU2 LBIST Unrecoverable FM Register (STCU2_LBUFM)

NOTE

The reset value is chip-specific; see the chapter that describes how modules are configured and connected.

The STCU2_LBUFM register defines the fault mapping of each LBIST in terms of Unrecoverable or Recoverable Fault.

The size of the register depends on the number of LBIST .

The R/W fields in this register are readable at any time. You can write to these fields when either of the following conditions is true:

- Off-line Self-Test phase is active
- On-line Self-Test phase is active and STCU2_CFG[WRP] = 0

Address: 0h base + 40h offset = 40h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0															
W																
Reset	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*
Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0										LBUFM5	LBUFM4	LBUFM3	LBUFM2	LBUFM1	LBUFM0
W																
Reset	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*

* Notes:

- The reset value is chip-specific; see the chapter that describes how modules are configured and connected.x = Undefined at reset.

STCU2_LBUFM field descriptions

Field	Description
0–25 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
26 LBUFM5	LBIST Unrecoverable Fault Mapping 0 Recoverable Fault mapping 1 Unrecoverable Fault mapping
27 LBUFM4	LBIST Unrecoverable Fault Mapping 0 Recoverable Fault mapping 1 Unrecoverable Fault mapping
28 LBUFM3	LBIST Unrecoverable Fault Mapping 0 Recoverable Fault mapping 1 Unrecoverable Fault mapping
29 LBUFM2	LBIST Unrecoverable Fault Mapping 0 Recoverable Fault mapping 1 Unrecoverable Fault mapping
30 LBUFM1	LBIST Unrecoverable Fault Mapping 0 Recoverable Fault mapping 1 Unrecoverable Fault mapping
31 LBUFM0	LBIST Unrecoverable Fault Mapping 0 Recoverable Fault mapping 1 Unrecoverable Fault mapping

63.9.16 STCU2 Off-Line MBIST Status Low Register (STCU2_MBSL)

The STCU2_MBSL register includes the results corresponding to the execution of the selected Off-Line MBIST in the range NMCUT = 0-31.

The size of the register depends on the number of BISTed RAMs/ROMs .

Register description

Address: 0h base + 44h offset = 44h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	MBS31	MBS30	MBS29	MBS28	MBS27	MBS26	MBS25	MBS24	MBS23	MBS22	MBS21	MBS20	MBS19	MBS18	MBS17	MBS16
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	MBS15	MBS14	MBS13	MBS12	MBS11	MBS10	MBS9	MBS8	MBS7	MBS6	MBS5	MBS4	MBS3	MBS2	MBS1	MBS0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

STCU2_MBSL field descriptions

Field	Description
0 MBS31	Off-Line status (NMCUT range = 0 to 31) of the selected MBIST NOTE: This bit is meaningful when the related bit of the STCU2_MBEL register reports the MBIST is finished. 0 Failed NMCUT BIST execution 1 No Fault detected during the NMCUT BIST execution
1 MBS30	Off-Line status (NMCUT range = 0 to 31) of the selected MBIST NOTE: This bit is meaningful when the related bit of the STCU2_MBEL register reports the MBIST is finished. 0 Failed NMCUT BIST execution 1 No Fault detected during the NMCUT BIST execution
2 MBS29	Off-Line status (NMCUT range = 0 to 31) of the selected MBIST NOTE: This bit is meaningful when the related bit of the STCU2_MBEL register reports the MBIST is finished. 0 Failed NMCUT BIST execution 1 No Fault detected during the NMCUT BIST execution

Table continues on the next page...

STCU2_MBSL field descriptions (continued)

Field	Description
3 MBS28	Off-Line status (NMCUT range = 0 to 31) of the selected MBIST NOTE: This bit is meaningful when the related bit of the STCU2_MBEL register reports the MBIST is finished. 0 Failed NMCUT BIST execution 1 No Fault detected during the NMCUT BIST execution
4 MBS27	Off-Line status (NMCUT range = 0 to 31) of the selected MBIST NOTE: This bit is meaningful when the related bit of the STCU2_MBEL register reports the MBIST is finished. 0 Failed NMCUT BIST execution 1 No Fault detected during the NMCUT BIST execution
5 MBS26	Off-Line status (NMCUT range = 0 to 31) of the selected MBIST NOTE: This bit is meaningful when the related bit of the STCU2_MBEL register reports the MBIST is finished. 0 Failed NMCUT BIST execution 1 No Fault detected during the NMCUT BIST execution
6 MBS25	Off-Line status (NMCUT range = 0 to 31) of the selected MBIST NOTE: This bit is meaningful when the related bit of the STCU2_MBEL register reports the MBIST is finished. 0 Failed NMCUT BIST execution 1 No Fault detected during the NMCUT BIST execution
7 MBS24	Off-Line status (NMCUT range = 0 to 31) of the selected MBIST NOTE: This bit is meaningful when the related bit of the STCU2_MBEL register reports the MBIST is finished. 0 Failed NMCUT BIST execution 1 No Fault detected during the NMCUT BIST execution
8 MBS23	Off-Line status (NMCUT range = 0 to 31) of the selected MBIST NOTE: This bit is meaningful when the related bit of the STCU2_MBEL register reports the MBIST is finished. 0 Failed NMCUT BIST execution 1 No Fault detected during the NMCUT BIST execution
9 MBS22	Off-Line status (NMCUT range = 0 to 31) of the selected MBIST NOTE: This bit is meaningful when the related bit of the STCU2_MBEL register reports the MBIST is finished. 0 Failed NMCUT BIST execution 1 No Fault detected during the NMCUT BIST execution
10 MBS21	Off-Line status (NMCUT range = 0 to 31) of the selected MBIST NOTE: This bit is meaningful when the related bit of the STCU2_MBEL register reports the MBIST is finished.

Table continues on the next page...

STCU2_MBSL field descriptions (continued)

Field	Description
	0 Failed NMCUT BIST execution 1 No Fault detected during the NMCUT BIST execution
11 MBS20	Off-Line status (NMCUT range = 0 to 31) of the selected MBIST NOTE: This bit is meaningful when the related bit of the STCU2_MBEL register reports the MBIST is finished. 0 Failed NMCUT BIST execution 1 No Fault detected during the NMCUT BIST execution
12 MBS19	Off-Line status (NMCUT range = 0 to 31) of the selected MBIST NOTE: This bit is meaningful when the related bit of the STCU2_MBEL register reports the MBIST is finished. 0 Failed NMCUT BIST execution 1 No Fault detected during the NMCUT BIST execution
13 MBS18	Off-Line status (NMCUT range = 0 to 31) of the selected MBIST NOTE: This bit is meaningful when the related bit of the STCU2_MBEL register reports the MBIST is finished. 0 Failed NMCUT BIST execution 1 No Fault detected during the NMCUT BIST execution
14 MBS17	Off-Line status (NMCUT range = 0 to 31) of the selected MBIST NOTE: This bit is meaningful when the related bit of the STCU2_MBEL register reports the MBIST is finished. 0 Failed NMCUT BIST execution 1 No Fault detected during the NMCUT BIST execution
15 MBS16	Off-Line status (NMCUT range = 0 to 31) of the selected MBIST NOTE: This bit is meaningful when the related bit of the STCU2_MBEL register reports the MBIST is finished. 0 Failed NMCUT BIST execution 1 No Fault detected during the NMCUT BIST execution
16 MBS15	Off-Line status (NMCUT range = 0 to 31) of the selected MBIST NOTE: This bit is meaningful when the related bit of the STCU2_MBEL register reports the MBIST is finished. 0 Failed NMCUT BIST execution 1 No Fault detected during the NMCUT BIST execution
17 MBS14	Off-Line status (NMCUT range = 0 to 31) of the selected MBIST NOTE: This bit is meaningful when the related bit of the STCU2_MBEL register reports the MBIST is finished. 0 Failed NMCUT BIST execution 1 No Fault detected during the NMCUT BIST execution
18 MBS13	Off-Line status (NMCUT range = 0 to 31) of the selected MBIST

Table continues on the next page...

STCU2_MBSL field descriptions (continued)

Field	Description
	NOTE: This bit is meaningful when the related bit of the STCU2_MBEL register reports the MBIST is finished. 0 Failed NMCUT BIST execution 1 No Fault detected during the NMCUT BIST execution
19 MBS12	Off-Line status (NMCUT range = 0 to 31) of the selected MBIST NOTE: This bit is meaningful when the related bit of the STCU2_MBEL register reports the MBIST is finished. 0 Failed NMCUT BIST execution 1 No Fault detected during the NMCUT BIST execution
20 MBS11	Off-Line status (NMCUT range = 0 to 31) of the selected MBIST NOTE: This bit is meaningful when the related bit of the STCU2_MBEL register reports the MBIST is finished. 0 Failed NMCUT BIST execution 1 No Fault detected during the NMCUT BIST execution
21 MBS10	Off-Line status (NMCUT range = 0 to 31) of the selected MBIST NOTE: This bit is meaningful when the related bit of the STCU2_MBEL register reports the MBIST is finished. 0 Failed NMCUT BIST execution 1 No Fault detected during the NMCUT BIST execution
22 MBS9	Off-Line status (NMCUT range = 0 to 31) of the selected MBIST NOTE: This bit is meaningful when the related bit of the STCU2_MBEL register reports the MBIST is finished. 0 Failed NMCUT BIST execution 1 No Fault detected during the NMCUT BIST execution
23 MBS8	Off-Line status (NMCUT range = 0 to 31) of the selected MBIST NOTE: This bit is meaningful when the related bit of the STCU2_MBEL register reports the MBIST is finished. 0 Failed NMCUT BIST execution 1 No Fault detected during the NMCUT BIST execution
24 MBS7	Off-Line status (NMCUT range = 0 to 31) of the selected MBIST NOTE: This bit is meaningful when the related bit of the STCU2_MBEL register reports the MBIST is finished. 0 Failed NMCUT BIST execution 1 No Fault detected during the NMCUT BIST execution
25 MBS6	Off-Line status (NMCUT range = 0 to 31) of the selected MBIST NOTE: This bit is meaningful when the related bit of the STCU2_MBEL register reports the MBIST is finished. 0 Failed NMCUT BIST execution 1 No Fault detected during the NMCUT BIST execution

Table continues on the next page...

STCU2_MBSL field descriptions (continued)

Field	Description
26 MBS5	Off-Line status (NMCUT range = 0 to 31) of the selected MBIST NOTE: This bit is meaningful when the related bit of the STCU2_MBEL register reports the MBIST is finished. 0 Failed NMCUT BIST execution 1 No Fault detected during the NMCUT BIST execution
27 MBS4	Off-Line status (NMCUT range = 0 to 31) of the selected MBIST NOTE: This bit is meaningful when the related bit of the STCU2_MBEL register reports the MBIST is finished. 0 Failed NMCUT BIST execution 1 No Fault detected during the NMCUT BIST execution
28 MBS3	Off-Line status (NMCUT range = 0 to 31) of the selected MBIST NOTE: This bit is meaningful when the related bit of the STCU2_MBEL register reports the MBIST is finished. 0 Failed NMCUT BIST execution 1 No Fault detected during the NMCUT BIST execution
29 MBS2	Off-Line status (NMCUT range = 0 to 31) of the selected MBIST NOTE: This bit is meaningful when the related bit of the STCU2_MBEL register reports the MBIST is finished. 0 Failed NMCUT BIST execution 1 No Fault detected during the NMCUT BIST execution
30 MBS1	Off-Line status (NMCUT range = 0 to 31) of the selected MBIST NOTE: This bit is meaningful when the related bit of the STCU2_MBEL register reports the MBIST is finished. 0 Failed NMCUT BIST execution 1 No Fault detected during the NMCUT BIST execution
31 MBS0	Off-Line status (NMCUT range = 0 to 31) of the selected MBIST NOTE: This bit is meaningful when the related bit of the STCU2_MBEL register reports the MBIST is finished. 0 Failed NMCUT BIST execution 1 No Fault detected during the NMCUT BIST execution

63.9.17 STCU2 Off-Line MBIST Status Medium Register (STCU2_MBSM)

The STCU2_MBSM register includes the results corresponding to the execution of the selected Off-Line MBIST in the range NMCUT = 32-63.

The size of the register depends on the number of BISTed RAMs/ROMs .

Address: 0h base + 48h offset = 48h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	MBS63	MBS62	MBS61	MBS60	MBS59	MBS58	MBS57	MBS56	MBS55	MBS54	MBS53	MBS52	MBS51	MBS50	MBS49	MBS48
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	MBS47	MBS46	MBS45	MBS44	MBS43	MBS42	MBS41	MBS40	MBS39	MBS38	MBS37	MBS36	MBS35	MBS34	MBS33	MBS32
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

STCU2_MBSM field descriptions

Field	Description
0 MBS63	MBSx: Off-Line status (NMCUT range = 32 to 63) of the selected MBIST NOTE: This bit is meaningful when the related bit of the STCU2_MBEM register reports the MBIST is finished). 0 Failed NMCUT BIST execution 1 No Fault detected during the NMCUT BIST execution
1 MBS62	MBSx: Off-Line status (NMCUT range = 32 to 63) of the selected MBIST NOTE: This bit is meaningful when the related bit of the STCU2_MBEM register reports the MBIST is finished). 0 Failed NMCUT BIST execution 1 No Fault detected during the NMCUT BIST execution
2 MBS61	MBSx: Off-Line status (NMCUT range = 32 to 63) of the selected MBIST NOTE: This bit is meaningful when the related bit of the STCU2_MBEM register reports the MBIST is finished). 0 Failed NMCUT BIST execution 1 No Fault detected during the NMCUT BIST execution

Table continues on the next page...

STCU2_MBSM field descriptions (continued)

Field	Description
3 MBS60	MBSx: Off-Line status (NMCUT range = 32 to 63) of the selected MBIST NOTE: This bit is meaningful when the related bit of the STCU2_MBEM register reports the MBIST is finished). 0 Failed NMCUT BIST execution 1 No Fault detected during the NMCUT BIST execution
4 MBS59	MBSx: Off-Line status (NMCUT range = 32 to 63) of the selected MBIST NOTE: This bit is meaningful when the related bit of the STCU2_MBEM register reports the MBIST is finished). 0 Failed NMCUT BIST execution 1 No Fault detected during the NMCUT BIST execution
5 MBS58	MBSx: Off-Line status (NMCUT range = 32 to 63) of the selected MBIST NOTE: This bit is meaningful when the related bit of the STCU2_MBEM register reports the MBIST is finished). 0 Failed NMCUT BIST execution 1 No Fault detected during the NMCUT BIST execution
6 MBS57	MBSx: Off-Line status (NMCUT range = 32 to 63) of the selected MBIST NOTE: This bit is meaningful when the related bit of the STCU2_MBEM register reports the MBIST is finished). 0 Failed NMCUT BIST execution 1 No Fault detected during the NMCUT BIST execution
7 MBS56	MBSx: Off-Line status (NMCUT range = 32 to 63) of the selected MBIST NOTE: This bit is meaningful when the related bit of the STCU2_MBEM register reports the MBIST is finished). 0 Failed NMCUT BIST execution 1 No Fault detected during the NMCUT BIST execution
8 MBS55	MBSx: Off-Line status (NMCUT range = 32 to 63) of the selected MBIST NOTE: This bit is meaningful when the related bit of the STCU2_MBEM register reports the MBIST is finished). 0 Failed NMCUT BIST execution 1 No Fault detected during the NMCUT BIST execution
9 MBS54	MBSx: Off-Line status (NMCUT range = 32 to 63) of the selected MBIST NOTE: This bit is meaningful when the related bit of the STCU2_MBEM register reports the MBIST is finished). 0 Failed NMCUT BIST execution 1 No Fault detected during the NMCUT BIST execution
10 MBS53	MBSx: Off-Line status (NMCUT range = 32 to 63) of the selected MBIST NOTE: This bit is meaningful when the related bit of the STCU2_MBEM register reports the MBIST is finished).

Table continues on the next page...

STCU2_MBSM field descriptions (continued)

Field	Description
	0 Failed NMCUT BIST execution 1 No Fault detected during the NMCUT BIST execution
11 MBS52	MBSx: Off-Line status (NMCUT range = 32 to 63) of the selected MBIST NOTE: This bit is meaningful when the related bit of the STCU2_MBEM register reports the MBIST is finished). 0 Failed NMCUT BIST execution 1 No Fault detected during the NMCUT BIST execution
12 MBS51	MBSx: Off-Line status (NMCUT range = 32 to 63) of the selected MBIST NOTE: This bit is meaningful when the related bit of the STCU2_MBEM register reports the MBIST is finished). 0 Failed NMCUT BIST execution 1 No Fault detected during the NMCUT BIST execution
13 MBS50	MBSx: Off-Line status (NMCUT range = 32 to 63) of the selected MBIST NOTE: This bit is meaningful when the related bit of the STCU2_MBEM register reports the MBIST is finished). 0 Failed NMCUT BIST execution 1 No Fault detected during the NMCUT BIST execution
14 MBS49	MBSx: Off-Line status (NMCUT range = 32 to 63) of the selected MBIST NOTE: This bit is meaningful when the related bit of the STCU2_MBEM register reports the MBIST is finished). 0 Failed NMCUT BIST execution 1 No Fault detected during the NMCUT BIST execution
15 MBS48	MBSx: Off-Line status (NMCUT range = 32 to 63) of the selected MBIST NOTE: This bit is meaningful when the related bit of the STCU2_MBEM register reports the MBIST is finished). 0 Failed NMCUT BIST execution 1 No Fault detected during the NMCUT BIST execution
16 MBS47	MBSx: Off-Line status (NMCUT range = 32 to 63) of the selected MBIST NOTE: This bit is meaningful when the related bit of the STCU2_MBEM register reports the MBIST is finished). 0 Failed NMCUT BIST execution 1 No Fault detected during the NMCUT BIST execution
17 MBS46	MBSx: Off-Line status (NMCUT range = 32 to 63) of the selected MBIST NOTE: This bit is meaningful when the related bit of the STCU2_MBEM register reports the MBIST is finished). 0 Failed NMCUT BIST execution 1 No Fault detected during the NMCUT BIST execution
18 MBS45	MBSx: Off-Line status (NMCUT range = 32 to 63) of the selected MBIST

Table continues on the next page...

STCU2_MBSM field descriptions (continued)

Field	Description
	NOTE: This bit is meaningful when the related bit of the STCU2_MBEM register reports the MBIST is finished). 0 Failed NMCUT BIST execution 1 No Fault detected during the NMCUT BIST execution
19 MBS44	MBSx: Off-Line status (NMCUT range = 32 to 63) of the selected MBIST NOTE: This bit is meaningful when the related bit of the STCU2_MBEM register reports the MBIST is finished). 0 Failed NMCUT BIST execution 1 No Fault detected during the NMCUT BIST execution
20 MBS43	MBSx: Off-Line status (NMCUT range = 32 to 63) of the selected MBIST NOTE: This bit is meaningful when the related bit of the STCU2_MBEM register reports the MBIST is finished). 0 Failed NMCUT BIST execution 1 No Fault detected during the NMCUT BIST execution
21 MBS42	MBSx: Off-Line status (NMCUT range = 32 to 63) of the selected MBIST NOTE: This bit is meaningful when the related bit of the STCU2_MBEM register reports the MBIST is finished). 0 Failed NMCUT BIST execution 1 No Fault detected during the NMCUT BIST execution
22 MBS41	MBSx: Off-Line status (NMCUT range = 32 to 63) of the selected MBIST NOTE: This bit is meaningful when the related bit of the STCU2_MBEM register reports the MBIST is finished). 0 Failed NMCUT BIST execution 1 No Fault detected during the NMCUT BIST execution
23 MBS40	MBSx: Off-Line status (NMCUT range = 32 to 63) of the selected MBIST NOTE: This bit is meaningful when the related bit of the STCU2_MBEM register reports the MBIST is finished). 0 Failed NMCUT BIST execution 1 No Fault detected during the NMCUT BIST execution
24 MBS39	MBSx: Off-Line status (NMCUT range = 32 to 63) of the selected MBIST NOTE: This bit is meaningful when the related bit of the STCU2_MBEM register reports the MBIST is finished). 0 Failed NMCUT BIST execution 1 No Fault detected during the NMCUT BIST execution
25 MBS38	MBSx: Off-Line status (NMCUT range = 32 to 63) of the selected MBIST NOTE: This bit is meaningful when the related bit of the STCU2_MBEM register reports the MBIST is finished). 0 Failed NMCUT BIST execution 1 No Fault detected during the NMCUT BIST execution

Table continues on the next page...

STCU2_MBSM field descriptions (continued)

Field	Description
26 MBS37	MBSx: Off-Line status (NMCUT range = 32 to 63) of the selected MBIST NOTE: This bit is meaningful when the related bit of the STCU2_MBEM register reports the MBIST is finished). 0 Failed NMCUT BIST execution 1 No Fault detected during the NMCUT BIST execution
27 MBS36	MBSx: Off-Line status (NMCUT range = 32 to 63) of the selected MBIST NOTE: This bit is meaningful when the related bit of the STCU2_MBEM register reports the MBIST is finished). 0 Failed NMCUT BIST execution 1 No Fault detected during the NMCUT BIST execution
28 MBS35	MBSx: Off-Line status (NMCUT range = 32 to 63) of the selected MBIST NOTE: This bit is meaningful when the related bit of the STCU2_MBEM register reports the MBIST is finished). 0 Failed NMCUT BIST execution 1 No Fault detected during the NMCUT BIST execution
29 MBS34	MBSx: Off-Line status (NMCUT range = 32 to 63) of the selected MBIST NOTE: This bit is meaningful when the related bit of the STCU2_MBEM register reports the MBIST is finished). 0 Failed NMCUT BIST execution 1 No Fault detected during the NMCUT BIST execution
30 MBS33	MBSx: Off-Line status (NMCUT range = 32 to 63) of the selected MBIST NOTE: This bit is meaningful when the related bit of the STCU2_MBEM register reports the MBIST is finished). 0 Failed NMCUT BIST execution 1 No Fault detected during the NMCUT BIST execution
31 MBS32	MBSx: Off-Line status (NMCUT range = 32 to 63) of the selected MBIST NOTE: This bit is meaningful when the related bit of the STCU2_MBEM register reports the MBIST is finished). 0 Failed NMCUT BIST execution 1 No Fault detected during the NMCUT BIST execution

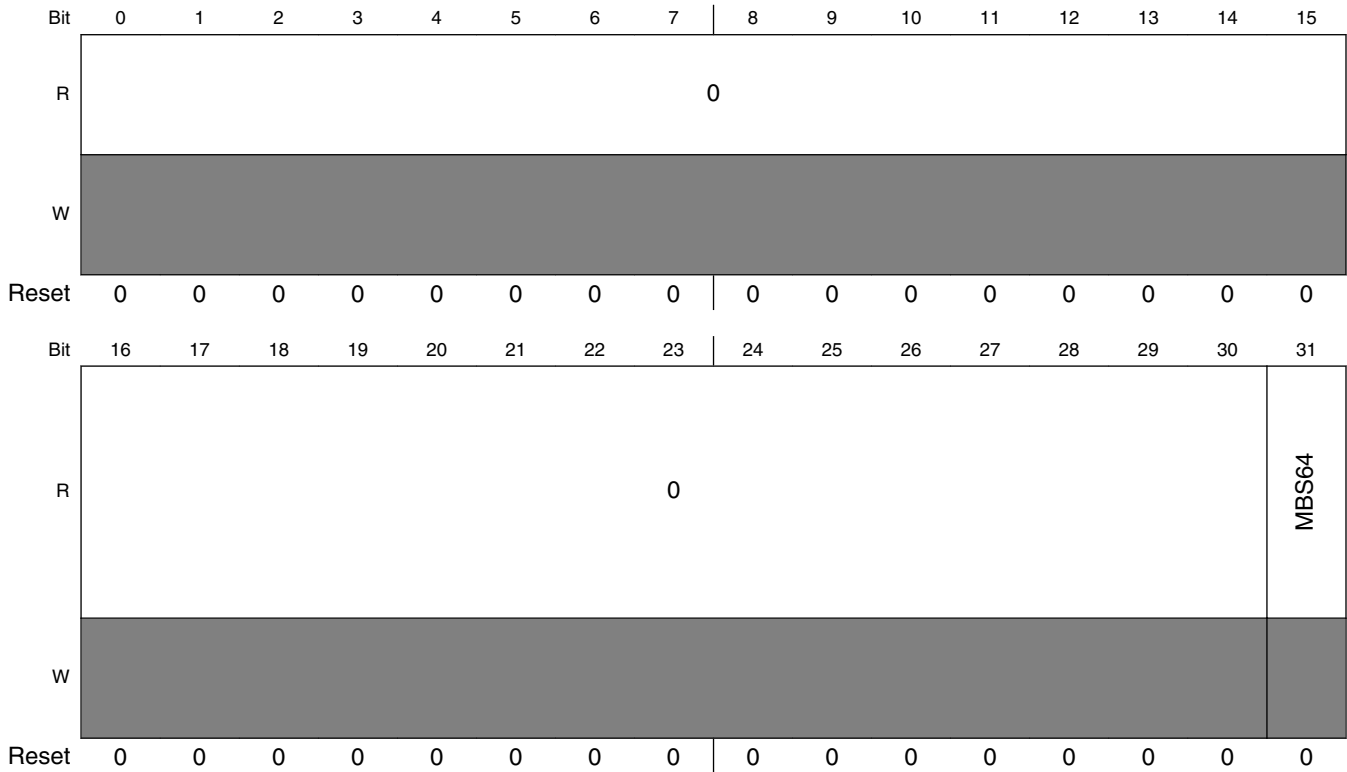
63.9.18 STCU2 Off-Line MBIST Status High Register (STCU2_MBSH)

The STCU2_MBSH register includes the results corresponding to the execution of the selected Off-Line MBIST in the range NMCUT = 64-95.

The size of the register depends on the number of BISTed RAMs/ROMs.

Register description

Address: 0h base + 4Ch offset = 4Ch



STCU2_MBSH field descriptions

Field	Description
0–30 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
31 MBS64	Off-Line status (NMCUT range = 64 to 95) of the selected MBIST NOTE: This bit is meaningful when the related bit of the STCU2_MBEH register reports the MBIST is finished). 0 Failed NMCUT BIST execution 1 No Fault detected during the NMCUT BIST execution

63.9.19 STCU2 Off-Line MBIST End Flag Low Register (STCU2_MBEL)

The STCU2_MBEL register includes the End Flag related to the execution of the selected Off-Line MBIST in the range NMCUT = 0-31.

The size of the register depends on the number of BISTed RAMs/ROMs.

Address: 0h base + 50h offset = 50h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	MBE31	MBE30	MBE29	MBE28	MBE27	MBE26	MBE25	MBE24	MBE23	MBE22	MBE21	MBE20	MBE19	MBE18	MBE17	MBE16
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	MBE15	MBE14	MBE13	MBE12	MBE11	MBE10	MBE9	MBE8	MBE7	MBE6	MBE5	MBE4	MBE3	MBE2	MBE1	MBE0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

STCU2_MBEL field descriptions

Field	Description
0 MBE31	Off-Line End status (0 to 31) of the selected MBIST 0 MBIST execution still ongoing 1 MBIST execution finished
1 MBE30	Off-Line End status (0 to 31) of the selected MBIST 0 MBIST execution still ongoing 1 MBIST execution finished
2 MBE29	Off-Line End status (0 to 31) of the selected MBIST 0 MBIST execution still ongoing 1 MBIST execution finished
3 MBE28	Off-Line End status (0 to 31) of the selected MBIST 0 MBIST execution still ongoing 1 MBIST execution finished
4 MBE27	Off-Line End status (0 to 31) of the selected MBIST 0 MBIST execution still ongoing 1 MBIST execution finished

Table continues on the next page...

STCU2_MBEL field descriptions (continued)

Field	Description
5 MBE26	Off-Line End status (0 to 31) of the selected MBIST 0 MBIST execution still ongoing 1 MBIST execution finished
6 MBE25	Off-Line End status (0 to 31) of the selected MBIST 0 MBIST execution still ongoing 1 MBIST execution finished
7 MBE24	Off-Line End status (0 to 31) of the selected MBIST 0 MBIST execution still ongoing 1 MBIST execution finished
8 MBE23	Off-Line End status (0 to 31) of the selected MBIST 0 MBIST execution still ongoing 1 MBIST execution finished
9 MBE22	Off-Line End status (0 to 31) of the selected MBIST 0 MBIST execution still ongoing 1 MBIST execution finished
10 MBE21	Off-Line End status (0 to 31) of the selected MBIST 0 MBIST execution still ongoing 1 MBIST execution finished
11 MBE20	Off-Line End status (0 to 31) of the selected MBIST 0 MBIST execution still ongoing 1 MBIST execution finished
12 MBE19	Off-Line End status (0 to 31) of the selected MBIST 0 MBIST execution still ongoing 1 MBIST execution finished
13 MBE18	Off-Line End status (0 to 31) of the selected MBIST 0 MBIST execution still ongoing 1 MBIST execution finished
14 MBE17	Off-Line End status (0 to 31) of the selected MBIST 0 MBIST execution still ongoing 1 MBIST execution finished
15 MBE16	Off-Line End status (0 to 31) of the selected MBIST 0 MBIST execution still ongoing 1 MBIST execution finished
16 MBE15	Off-Line End status (0 to 31) of the selected MBIST 0 MBIST execution still ongoing 1 MBIST execution finished

Table continues on the next page...

STCU2_MBEL field descriptions (continued)

Field	Description
17 MBE14	Off-Line End status (0 to 31) of the selected MBIST 0 MBIST execution still ongoing 1 MBIST execution finished
18 MBE13	Off-Line End status (0 to 31) of the selected MBIST 0 MBIST execution still ongoing 1 MBIST execution finished
19 MBE12	Off-Line End status (0 to 31) of the selected MBIST 0 MBIST execution still ongoing 1 MBIST execution finished
20 MBE11	Off-Line End status (0 to 31) of the selected MBIST 0 MBIST execution still ongoing 1 MBIST execution finished
21 MBE10	Off-Line End status (0 to 31) of the selected MBIST 0 MBIST execution still ongoing 1 MBIST execution finished
22 MBE9	Off-Line End status (0 to 31) of the selected MBIST 0 MBIST execution still ongoing 1 MBIST execution finished
23 MBE8	Off-Line End status (0 to 31) of the selected MBIST 0 MBIST execution still ongoing 1 MBIST execution finished
24 MBE7	Off-Line End status (0 to 31) of the selected MBIST 0 MBIST execution still ongoing 1 MBIST execution finished
25 MBE6	Off-Line End status (0 to 31) of the selected MBIST 0 MBIST execution still ongoing 1 MBIST execution finished
26 MBE5	Off-Line End status (0 to 31) of the selected MBIST 0 MBIST execution still ongoing 1 MBIST execution finished
27 MBE4	Off-Line End status (0 to 31) of the selected MBIST 0 MBIST execution still ongoing 1 MBIST execution finished
28 MBE3	Off-Line End status (0 to 31) of the selected MBIST 0 MBIST execution still ongoing 1 MBIST execution finished

Table continues on the next page...

STCU2_MBEL field descriptions (continued)

Field	Description
29 MBE2	Off-Line End status (0 to 31) of the selected MBIST 0 MBIST execution still ongoing 1 MBIST execution finished
30 MBE1	Off-Line End status (0 to 31) of the selected MBIST 0 MBIST execution still ongoing 1 MBIST execution finished
31 MBE0	Off-Line End status (0 to 31) of the selected MBIST 0 MBIST execution still ongoing 1 MBIST execution finished

63.9.20 STCU2 Off-Line MBIST End Flag Medium Register (STCU2_MBEM)

The STCU2_MBEM register includes the End Flag related to the execution of the selected Off-Line MBIST in the range NMCUT = 32-63.

The size of the register depends on the number of BISTed RAMs/ROMs.

Address: 0h base + 54h offset = 54h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	MBE63	MBE62	MBE61	MBE60	MBE59	MBE58	MBE57	MBE56	MBE55	MBE54	MBE53	MBE52	MBE51	MBE50	MBE49	MBE48
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	MBE47	MBE46	MBE45	MBE44	MBE43	MBE42	MBE41	MBE40	MBE39	MBE38	MBE37	MBE36	MBE35	MBE34	MBE33	MBE32
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

STCU2_MBEM field descriptions

Field	Description
0 MBE63	Off-Line End status (32 to 63) of the selected MBIST 0 MBIST execution still ongoing 1 MBIST execution finished
1 MBE62	Off-Line End status (32 to 63) of the selected MBIST 0 MBIST execution still ongoing 1 MBIST execution finished
2 MBE61	Off-Line End status (32 to 63) of the selected MBIST 0 MBIST execution still ongoing 1 MBIST execution finished
3 MBE60	Off-Line End status (32 to 63) of the selected MBIST 0 MBIST execution still ongoing 1 MBIST execution finished
4 MBE59	Off-Line End status (32 to 63) of the selected MBIST 0 MBIST execution still ongoing 1 MBIST execution finished
5 MBE58	Off-Line End status (32 to 63) of the selected MBIST 0 MBIST execution still ongoing 1 MBIST execution finished
6 MBE57	Off-Line End status (32 to 63) of the selected MBIST 0 MBIST execution still ongoing 1 MBIST execution finished
7 MBE56	Off-Line End status (32 to 63) of the selected MBIST 0 MBIST execution still ongoing 1 MBIST execution finished
8 MBE55	Off-Line End status (32 to 63) of the selected MBIST

Table continues on the next page...

STCU2_MBEM field descriptions (continued)

Field	Description
	0 MBIST execution still ongoing 1 MBIST execution finished
9 MBE54	Off-Line End status (32 to 63) of the selected MBIST 0 MBIST execution still ongoing 1 MBIST execution finished
10 MBE53	Off-Line End status (32 to 63) of the selected MBIST 0 MBIST execution still ongoing 1 MBIST execution finished
11 MBE52	Off-Line End status (32 to 63) of the selected MBIST 0 MBIST execution still ongoing 1 MBIST execution finished
12 MBE51	Off-Line End status (32 to 63) of the selected MBIST 0 MBIST execution still ongoing 1 MBIST execution finished
13 MBE50	Off-Line End status (32 to 63) of the selected MBIST 0 MBIST execution still ongoing 1 MBIST execution finished
14 MBE49	Off-Line End status (32 to 63) of the selected MBIST 0 MBIST execution still ongoing 1 MBIST execution finished
15 MBE48	Off-Line End status (32 to 63) of the selected MBIST 0 MBIST execution still ongoing 1 MBIST execution finished
16 MBE47	Off-Line End status (32 to 63) of the selected MBIST 0 MBIST execution still ongoing 1 MBIST execution finished
17 MBE46	Off-Line End status (32 to 63) of the selected MBIST 0 MBIST execution still ongoing 1 MBIST execution finished
18 MBE45	Off-Line End status (32 to 63) of the selected MBIST 0 MBIST execution still ongoing 1 MBIST execution finished
19 MBE44	Off-Line End status (32 to 63) of the selected MBIST 0 MBIST execution still ongoing 1 MBIST execution finished
20 MBE43	Off-Line End status (32 to 63) of the selected MBIST

Table continues on the next page...

STCU2_MBEM field descriptions (continued)

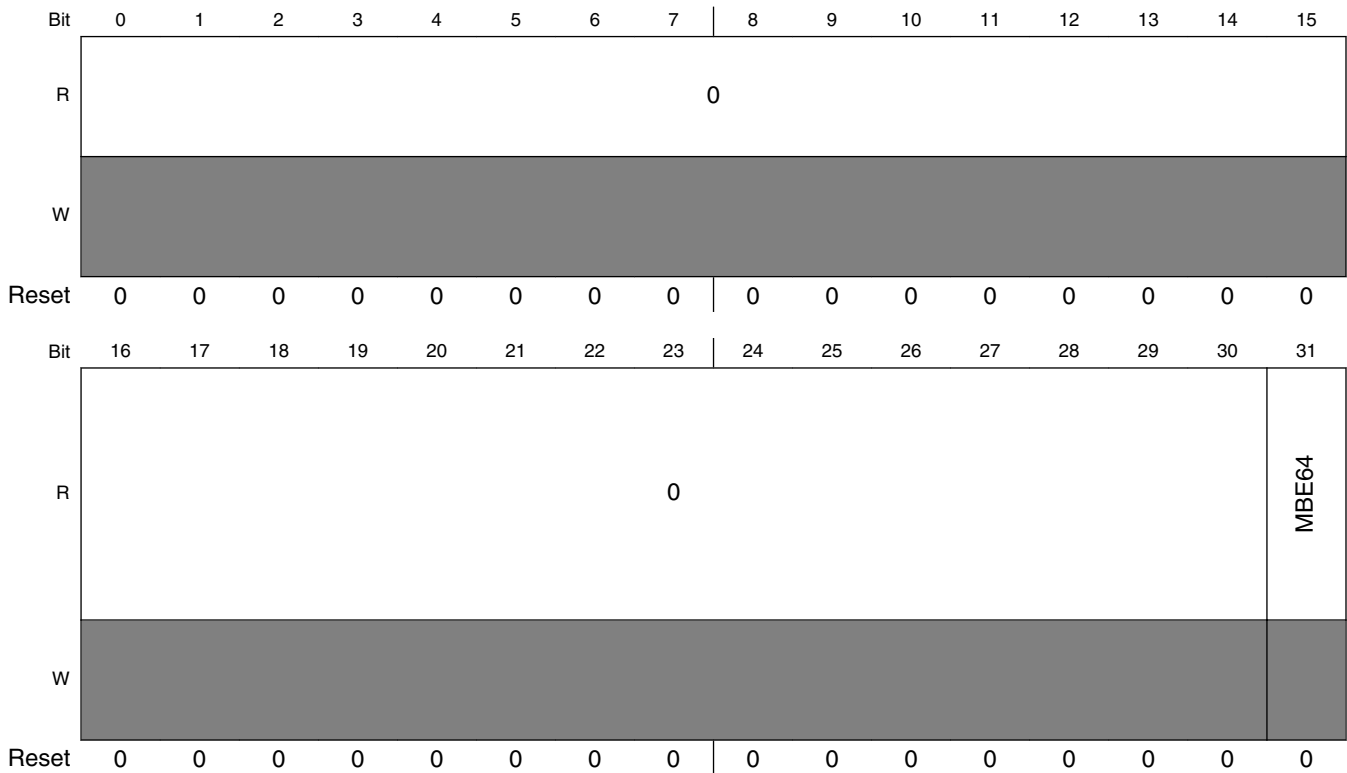
Field	Description
	0 MBIST execution still ongoing 1 MBIST execution finished
21 MBE42	Off-Line End status (32 to 63) of the selected MBIST 0 MBIST execution still ongoing 1 MBIST execution finished
22 MBE41	Off-Line End status (32 to 63) of the selected MBIST 0 MBIST execution still ongoing 1 MBIST execution finished
23 MBE40	Off-Line End status (32 to 63) of the selected MBIST 0 MBIST execution still ongoing 1 MBIST execution finished
24 MBE39	Off-Line End status (32 to 63) of the selected MBIST 0 MBIST execution still ongoing 1 MBIST execution finished
25 MBE38	Off-Line End status (32 to 63) of the selected MBIST 0 MBIST execution still ongoing 1 MBIST execution finished
26 MBE37	Off-Line End status (32 to 63) of the selected MBIST 0 MBIST execution still ongoing 1 MBIST execution finished
27 MBE36	Off-Line End status (32 to 63) of the selected MBIST 0 MBIST execution still ongoing 1 MBIST execution finished
28 MBE35	Off-Line End status (32 to 63) of the selected MBIST 0 MBIST execution still ongoing 1 MBIST execution finished
29 MBE34	Off-Line End status (32 to 63) of the selected MBIST 0 MBIST execution still ongoing 1 MBIST execution finished
30 MBE33	Off-Line End status (32 to 63) of the selected MBIST 0 MBIST execution still ongoing 1 MBIST execution finished
31 MBE32	Off-Line End status (32 to 63) of the selected MBIST 0 MBIST execution still ongoing 1 MBIST execution finished

63.9.21 STCU2 Off-Line MBIST End Flag High Register (STCU2_MBEH)

The STCU2_MBEH register includes the End Flag related to the execution of the selected Off-Line MBIST in the range NMCUT = 64..95.

The size of the register depends on the number of BISTed RAMs/ROMs.

Address: 0h base + 58h offset = 58h



STCU2_MBEH field descriptions

Field	Description
0–30 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
31 MBE64	MBEx: Off-Line End status (64 to 95) of the selected MBIST 0 MBIST execution still ongoing 1 MBIST execution finished

63.9.22 STCU2 On-Line MBIST Status Low Register (STCU2_MBSLSW)

The STCU2_MBSLSW register includes the results corresponding to the execution of the selected On-Line MBIST in the range NMCUT = 0-31.

The size of the register depends on the number of BISTed RAMs/ROMs.

The content of this register is initialized to its reset value as soon as STCU2_RUNSW[RUNSW] = 1.

Address: 0h base + 5Ch offset = 5Ch

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	MBSSW31	MBSSW30	MBSSW29	MBSSW28	MBSSW27	MBSSW26	MBSSW25	MBSSW24	MBSSW23	MBSSW22	MBSSW21	MBSSW20	MBSSW19	MBSSW18	MBSSW17	MBSSW16
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	MBSSW15	MBSSW14	MBSSW13	MBSSW12	MBSSW11	MBSSW10	MBSSW9	MBSSW8	MBSSW7	MBSSW6	MBSSW5	MBSSW4	MBSSW3	MBSSW2	MBSSW1	MBSSW0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

STCU2_MBSLSW field descriptions

Field	Description
0 MBSSW31	On-Line status (NMCUT range = 0 to 31) of the selected MBIST

Table continues on the next page...

STCU2_MBSLSW field descriptions (continued)

Field	Description
	0 Failed NMCUT BIST execution 1 No Fault detected during the NMCUT BIST execution
1 MBSSW30	On-Line status (NMCUT range = 0 to 31) of the selected MBIST 0 Failed NMCUT BIST execution 1 No Fault detected during the NMCUT BIST execution
2 MBSSW29	On-Line status (NMCUT range = 0 to 31) of the selected MBIST 0 Failed NMCUT BIST execution 1 No Fault detected during the NMCUT BIST execution
3 MBSSW28	On-Line status (NMCUT range = 0 to 31) of the selected MBIST 0 Failed NMCUT BIST execution 1 No Fault detected during the NMCUT BIST execution
4 MBSSW27	On-Line status (NMCUT range = 0 to 31) of the selected MBIST 0 Failed NMCUT BIST execution 1 No Fault detected during the NMCUT BIST execution
5 MBSSW26	On-Line status (NMCUT range = 0 to 31) of the selected MBIST 0 Failed NMCUT BIST execution 1 No Fault detected during the NMCUT BIST execution
6 MBSSW25	On-Line status (NMCUT range = 0 to 31) of the selected MBIST 0 Failed NMCUT BIST execution 1 No Fault detected during the NMCUT BIST execution
7 MBSSW24	On-Line status (NMCUT range = 0 to 31) of the selected MBIST 0 Failed NMCUT BIST execution 1 No Fault detected during the NMCUT BIST execution
8 MBSSW23	On-Line status (NMCUT range = 0 to 31) of the selected MBIST 0 Failed NMCUT BIST execution 1 No Fault detected during the NMCUT BIST execution
9 MBSSW22	On-Line status (NMCUT range = 0 to 31) of the selected MBIST 0 Failed NMCUT BIST execution 1 No Fault detected during the NMCUT BIST execution
10 MBSSW21	On-Line status (NMCUT range = 0 to 31) of the selected MBIST 0 Failed NMCUT BIST execution 1 No Fault detected during the NMCUT BIST execution
11 MBSSW20	On-Line status (NMCUT range = 0 to 31) of the selected MBIST 0 Failed NMCUT BIST execution 1 No Fault detected during the NMCUT BIST execution
12 MBSSW19	On-Line status (NMCUT range = 0 to 31) of the selected MBIST

Table continues on the next page...

STCU2_MBSLSW field descriptions (continued)

Field	Description
	0 Failed NMCUT BIST execution 1 No Fault detected during the NMCUT BIST execution
13 MBSSW18	On-Line status (NMCUT range = 0 to 31) of the selected MBIST 0 Failed NMCUT BIST execution 1 No Fault detected during the NMCUT BIST execution
14 MBSSW17	On-Line status (NMCUT range = 0 to 31) of the selected MBIST 0 Failed NMCUT BIST execution 1 No Fault detected during the NMCUT BIST execution
15 MBSSW16	On-Line status (NMCUT range = 0 to 31) of the selected MBIST 0 Failed NMCUT BIST execution 1 No Fault detected during the NMCUT BIST execution
16 MBSSW15	On-Line status (NMCUT range = 0 to 31) of the selected MBIST 0 Failed NMCUT BIST execution 1 No Fault detected during the NMCUT BIST execution
17 MBSSW14	On-Line status (NMCUT range = 0 to 31) of the selected MBIST 0 Failed NMCUT BIST execution 1 No Fault detected during the NMCUT BIST execution
18 MBSSW13	On-Line status (NMCUT range = 0 to 31) of the selected MBIST 0 Failed NMCUT BIST execution 1 No Fault detected during the NMCUT BIST execution
19 MBSSW12	On-Line status (NMCUT range = 0 to 31) of the selected MBIST 0 Failed NMCUT BIST execution 1 No Fault detected during the NMCUT BIST execution
20 MBSSW11	On-Line status (NMCUT range = 0 to 31) of the selected MBIST 0 Failed NMCUT BIST execution 1 No Fault detected during the NMCUT BIST execution
21 MBSSW10	On-Line status (NMCUT range = 0 to 31) of the selected MBIST 0 Failed NMCUT BIST execution 1 No Fault detected during the NMCUT BIST execution
22 MBSSW9	On-Line status (NMCUT range = 0 to 31) of the selected MBIST 0 Failed NMCUT BIST execution 1 No Fault detected during the NMCUT BIST execution
23 MBSSW8	On-Line status (NMCUT range = 0 to 31) of the selected MBIST 0 Failed NMCUT BIST execution 1 No Fault detected during the NMCUT BIST execution
24 MBSSW7	On-Line status (NMCUT range = 0 to 31) of the selected MBIST

Table continues on the next page...

STCU2_MBSLSW field descriptions (continued)

Field	Description
	0 Failed NMCUT BIST execution 1 No Fault detected during the NMCUT BIST execution
25 MBSSW6	On-Line status (NMCUT range = 0 to 31) of the selected MBIST 0 Failed NMCUT BIST execution 1 No Fault detected during the NMCUT BIST execution
26 MBSSW5	On-Line status (NMCUT range = 0 to 31) of the selected MBIST 0 Failed NMCUT BIST execution 1 No Fault detected during the NMCUT BIST execution
27 MBSSW4	On-Line status (NMCUT range = 0 to 31) of the selected MBIST 0 Failed NMCUT BIST execution 1 No Fault detected during the NMCUT BIST execution
28 MBSSW3	On-Line status (NMCUT range = 0 to 31) of the selected MBIST 0 Failed NMCUT BIST execution 1 No Fault detected during the NMCUT BIST execution
29 MBSSW2	On-Line status (NMCUT range = 0 to 31) of the selected MBIST 0 Failed NMCUT BIST execution 1 No Fault detected during the NMCUT BIST execution
30 MBSSW1	On-Line status (NMCUT range = 0 to 31) of the selected MBIST 0 Failed NMCUT BIST execution 1 No Fault detected during the NMCUT BIST execution
31 MBSSW0	On-Line status (NMCUT range = 0 to 31) of the selected MBIST 0 Failed NMCUT BIST execution 1 No Fault detected during the NMCUT BIST execution

63.9.23 STCU2 On-Line MBIST Status Medium Register (STCU2_MBSMSW)

The STCU2_MBSMSW register includes the results corresponding to the execution of the selected On-Line MBIST in the range NMCUT = 32..63.

The size of the register depends on the number of BISTed RAMs/ROMs.

The content of this register is initialized to its reset value as soon as STCU2_RUNSW[RUNSW] = 1.

Address: 0h base + 60h offset = 60h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	MBSSW63	MBSSW62	MBSSW61	MBSSW60	MBSSW59	MBSSW58	MBSSW57	MBSSW56	MBSSW55	MBSSW54	MBSSW53	MBSSW52	MBSSW51	MBSSW50	MBSSW49	MBSSW48
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	MBSSW47	MBSSW46	MBSSW45	MBSSW44	MBSSW43	MBSSW42	MBSSW41	MBSSW40	MBSSW39	MBSSW38	MBSSW37	MBSSW36	MBSSW35	MBSSW34	MBSSW33	MBSSW32
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

STCU2_MBSMSW field descriptions

Field	Description
0 MBSSW63	MBSSWx: On-Line status (NMCUT range = 32 to 63) of the selected MBIST 0 Failed NMCUT BIST execution 1 No Fault detected during the NMCUT BIST execution
1 MBSSW62	MBSSWx: On-Line status (NMCUT range = 32 to 63) of the selected MBIST 0 Failed NMCUT BIST execution 1 No Fault detected during the NMCUT BIST execution
2 MBSSW61	MBSSWx: On-Line status (NMCUT range = 32 to 63) of the selected MBIST 0 Failed NMCUT BIST execution 1 No Fault detected during the NMCUT BIST execution
3 MBSSW60	MBSSWx: On-Line status (NMCUT range = 32 to 63) of the selected MBIST

Table continues on the next page...

STCU2_MBSMSW field descriptions (continued)

Field	Description
	0 Failed NMCUT BIST execution 1 No Fault detected during the NMCUT BIST execution
4 MBSSW59	MBSSWx: On-Line status (NMCUT range = 32 to 63) of the selected MBIST 0 Failed NMCUT BIST execution 1 No Fault detected during the NMCUT BIST execution
5 MBSSW58	MBSSWx: On-Line status (NMCUT range = 32 to 63) of the selected MBIST 0 Failed NMCUT BIST execution 1 No Fault detected during the NMCUT BIST execution
6 MBSSW57	MBSSWx: On-Line status (NMCUT range = 32 to 63) of the selected MBIST 0 Failed NMCUT BIST execution 1 No Fault detected during the NMCUT BIST execution
7 MBSSW56	MBSSWx: On-Line status (NMCUT range = 32 to 63) of the selected MBIST 0 Failed NMCUT BIST execution 1 No Fault detected during the NMCUT BIST execution
8 MBSSW55	MBSSWx: On-Line status (NMCUT range = 32 to 63) of the selected MBIST 0 Failed NMCUT BIST execution 1 No Fault detected during the NMCUT BIST execution
9 MBSSW54	MBSSWx: On-Line status (NMCUT range = 32 to 63) of the selected MBIST 0 Failed NMCUT BIST execution 1 No Fault detected during the NMCUT BIST execution
10 MBSSW53	MBSSWx: On-Line status (NMCUT range = 32 to 63) of the selected MBIST 0 Failed NMCUT BIST execution 1 No Fault detected during the NMCUT BIST execution
11 MBSSW52	MBSSWx: On-Line status (NMCUT range = 32 to 63) of the selected MBIST 0 Failed NMCUT BIST execution 1 No Fault detected during the NMCUT BIST execution
12 MBSSW51	MBSSWx: On-Line status (NMCUT range = 32 to 63) of the selected MBIST 0 Failed NMCUT BIST execution 1 No Fault detected during the NMCUT BIST execution
13 MBSSW50	MBSSWx: On-Line status (NMCUT range = 32 to 63) of the selected MBIST 0 Failed NMCUT BIST execution 1 No Fault detected during the NMCUT BIST execution
14 MBSSW49	MBSSWx: On-Line status (NMCUT range = 32 to 63) of the selected MBIST 0 Failed NMCUT BIST execution 1 No Fault detected during the NMCUT BIST execution
15 MBSSW48	MBSSWx: On-Line status (NMCUT range = 32 to 63) of the selected MBIST

Table continues on the next page...

STCU2_MBSMSW field descriptions (continued)

Field	Description
	0 Failed NMCUT BIST execution 1 No Fault detected during the NMCUT BIST execution
16 MBSSW47	MBSSWx: On-Line status (NMCUT range = 32 to 63) of the selected MBIST 0 Failed NMCUT BIST execution 1 No Fault detected during the NMCUT BIST execution
17 MBSSW46	MBSSWx: On-Line status (NMCUT range = 32 to 63) of the selected MBIST 0 Failed NMCUT BIST execution 1 No Fault detected during the NMCUT BIST execution
18 MBSSW45	MBSSWx: On-Line status (NMCUT range = 32 to 63) of the selected MBIST 0 Failed NMCUT BIST execution 1 No Fault detected during the NMCUT BIST execution
19 MBSSW44	MBSSWx: On-Line status (NMCUT range = 32 to 63) of the selected MBIST 0 Failed NMCUT BIST execution 1 No Fault detected during the NMCUT BIST execution
20 MBSSW43	MBSSWx: On-Line status (NMCUT range = 32 to 63) of the selected MBIST 0 Failed NMCUT BIST execution 1 No Fault detected during the NMCUT BIST execution
21 MBSSW42	MBSSWx: On-Line status (NMCUT range = 32 to 63) of the selected MBIST 0 Failed NMCUT BIST execution 1 No Fault detected during the NMCUT BIST execution
22 MBSSW41	MBSSWx: On-Line status (NMCUT range = 32 to 63) of the selected MBIST 0 Failed NMCUT BIST execution 1 No Fault detected during the NMCUT BIST execution
23 MBSSW40	MBSSWx: On-Line status (NMCUT range = 32 to 63) of the selected MBIST 0 Failed NMCUT BIST execution 1 No Fault detected during the NMCUT BIST execution
24 MBSSW39	MBSSWx: On-Line status (NMCUT range = 32 to 63) of the selected MBIST 0 Failed NMCUT BIST execution 1 No Fault detected during the NMCUT BIST execution
25 MBSSW38	MBSSWx: On-Line status (NMCUT range = 32 to 63) of the selected MBIST 0 Failed NMCUT BIST execution 1 No Fault detected during the NMCUT BIST execution
26 MBSSW37	MBSSWx: On-Line status (NMCUT range = 32 to 63) of the selected MBIST 0 Failed NMCUT BIST execution 1 No Fault detected during the NMCUT BIST execution
27 MBSSW36	MBSSWx: On-Line status (NMCUT range = 32 to 63) of the selected MBIST

Table continues on the next page...

STCU2_MBSMSW field descriptions (continued)

Field	Description
	0 Failed NMCUT BIST execution 1 No Fault detected during the NMCUT BIST execution
28 MBSSW35	MBSSWx: On-Line status (NMCUT range = 32 to 63) of the selected MBIST 0 Failed NMCUT BIST execution 1 No Fault detected during the NMCUT BIST execution
29 MBSSW34	MBSSWx: On-Line status (NMCUT range = 32 to 63) of the selected MBIST 0 Failed NMCUT BIST execution 1 No Fault detected during the NMCUT BIST execution
30 MBSSW33	MBSSWx: On-Line status (NMCUT range = 32 to 63) of the selected MBIST 0 Failed NMCUT BIST execution 1 No Fault detected during the NMCUT BIST execution
31 MBSSW32	MBSSWx: On-Line status (NMCUT range = 32 to 63) of the selected MBIST 0 Failed NMCUT BIST execution 1 No Fault detected during the NMCUT BIST execution

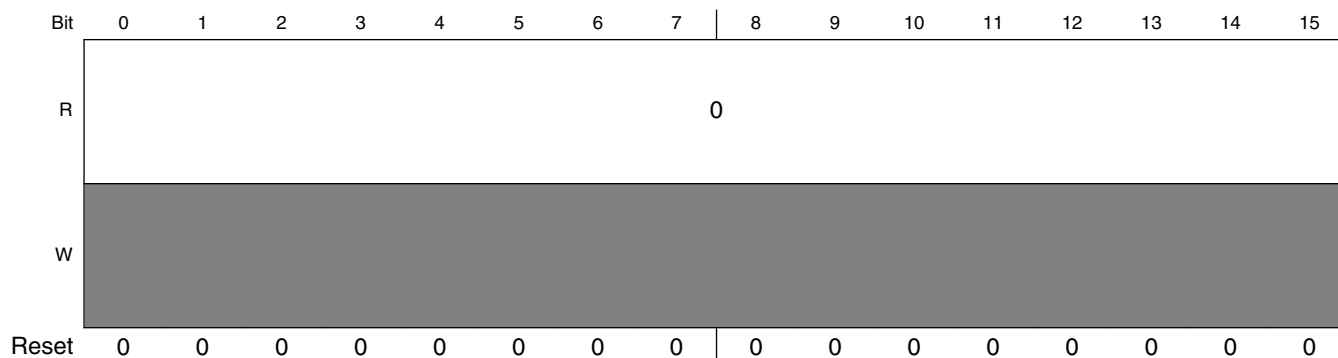
63.9.24 STCU2 On-Line MBIST Status High Register (STCU2_MBSHSW)

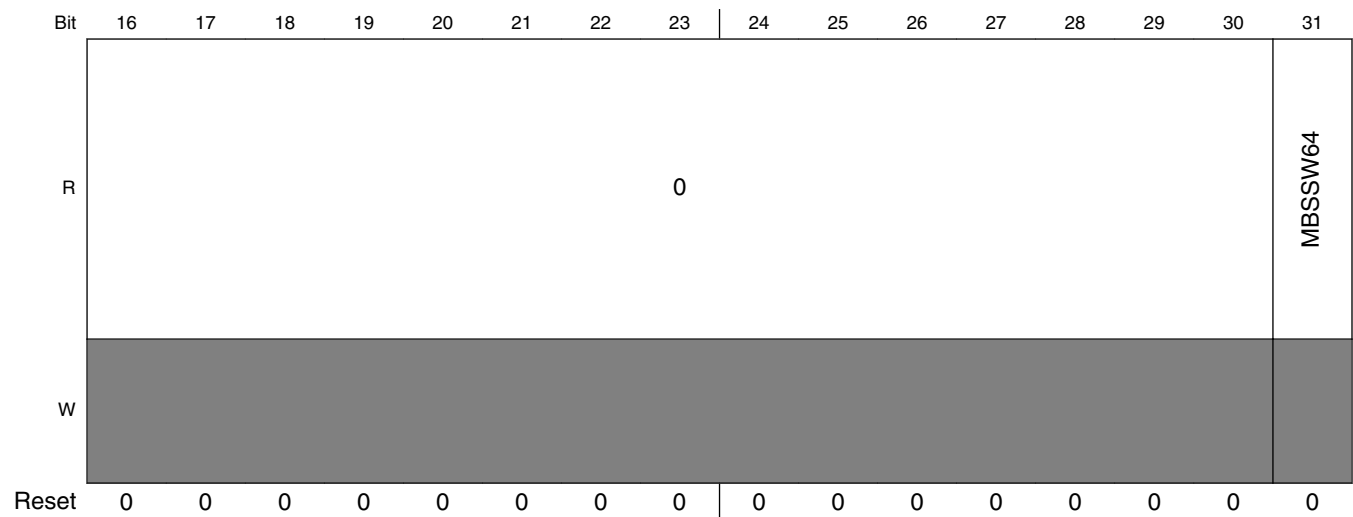
The STCU2_MBSHSW register includes the results corresponding to the execution of the selected On-Line MBIST in the range NMCUT = 64..95.

The size of the register depends on the number of BISTed RAMs/ROMs.

The content of this register is initialized to its reset value as soon as STCU2_RUNSW[RUNSW] = 1.

Address: 0h base + 64h offset = 64h





STCU2_MBSHSW field descriptions

Field	Description
0–30 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
31 MBSSW64	MBSSWx: On-Line status (NMCUT range = 64 to 95) of the selected MBIST 0 Failed NMCUT BIST execution 1 No Fault detected during the NMCUT BIST execution

63.9.25 STCU2 On-Line MBIST End Flag Low Register (STCU2_MBELSW)

The STCU2_MBELSW register includes the End Flag related to the execution of the selected On-Line MBIST in the range NMCUT = 0-31.

The size of the register depends on the number of BISTed RAMs/ROMs.

The content of this register is initialized to its reset value as soon as STCU2_RUNSW[RUNSW] = 1.

Register description

Address: 0h base + 68h offset = 68h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	MBESW31	MBESW30	MBESW29	MBESW28	MBESW27	MBESW26	MBESW25	MBESW24	MBESW23	MBESW22	MBESW21	MBESW20	MBESW19	MBESW18	MBESW17	MBESW16
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	MBESW15	MBESW14	MBESW13	MBESW12	MBESW11	MBESW10	MBESW9	MBESW8	MBESW7	MBESW6	MBESW5	MBESW4	MBESW3	MBESW2	MBESW1	MBESW0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

STCU2_MBELSW field descriptions

Field	Description
0 MBESW31	MBESWx: On-Line End status (0 to 31) of the selected MBIST 0 MBIST execution still ongoing 1 MBIST execution finished
1 MBESW30	MBESWx: On-Line End status (0 to 31) of the selected MBIST 0 MBIST execution still ongoing 1 MBIST execution finished
2 MBESW29	MBESWx: On-Line End status (0 to 31) of the selected MBIST 0 MBIST execution still ongoing 1 MBIST execution finished
3 MBESW28	MBESWx: On-Line End status (0 to 31) of the selected MBIST

Table continues on the next page...

STCU2_MBESW field descriptions (continued)

Field	Description
	0 MBIST execution still ongoing 1 MBIST execution finished
4 MBESW27	MBESWx: On-Line End status (0 to 31) of the selected MBIST 0 MBIST execution still ongoing 1 MBIST execution finished
5 MBESW26	MBESWx: On-Line End status (0 to 31) of the selected MBIST 0 MBIST execution still ongoing 1 MBIST execution finished
6 MBESW25	MBESWx: On-Line End status (0 to 31) of the selected MBIST 0 MBIST execution still ongoing 1 MBIST execution finished
7 MBESW24	MBESWx: On-Line End status (0 to 31) of the selected MBIST 0 MBIST execution still ongoing 1 MBIST execution finished
8 MBESW23	MBESWx: On-Line End status (0 to 31) of the selected MBIST 0 MBIST execution still ongoing 1 MBIST execution finished
9 MBESW22	MBESWx: On-Line End status (0 to 31) of the selected MBIST 0 MBIST execution still ongoing 1 MBIST execution finished
10 MBESW21	MBESWx: On-Line End status (0 to 31) of the selected MBIST 0 MBIST execution still ongoing 1 MBIST execution finished
11 MBESW20	MBESWx: On-Line End status (0 to 31) of the selected MBIST 0 MBIST execution still ongoing 1 MBIST execution finished
12 MBESW19	MBESWx: On-Line End status (0 to 31) of the selected MBIST 0 MBIST execution still ongoing 1 MBIST execution finished
13 MBESW18	MBESWx: On-Line End status (0 to 31) of the selected MBIST 0 MBIST execution still ongoing 1 MBIST execution finished
14 MBESW17	MBESWx: On-Line End status (0 to 31) of the selected MBIST 0 MBIST execution still ongoing 1 MBIST execution finished
15 MBESW16	MBESWx: On-Line End status (0 to 31) of the selected MBIST

Table continues on the next page...

STCU2_MBESW field descriptions (continued)

Field	Description
	0 MBIST execution still ongoing 1 MBIST execution finished
16 MBESW15	MBESWx: On-Line End status (0 to 31) of the selected MBIST 0 MBIST execution still ongoing 1 MBIST execution finished
17 MBESW14	MBESWx: On-Line End status (0 to 31) of the selected MBIST 0 MBIST execution still ongoing 1 MBIST execution finished
18 MBESW13	MBESWx: On-Line End status (0 to 31) of the selected MBIST 0 MBIST execution still ongoing 1 MBIST execution finished
19 MBESW12	MBESWx: On-Line End status (0 to 31) of the selected MBIST 0 MBIST execution still ongoing 1 MBIST execution finished
20 MBESW11	MBESWx: On-Line End status (0 to 31) of the selected MBIST 0 MBIST execution still ongoing 1 MBIST execution finished
21 MBESW10	MBESWx: On-Line End status (0 to 31) of the selected MBIST 0 MBIST execution still ongoing 1 MBIST execution finished
22 MBESW9	MBESWx: On-Line End status (0 to 31) of the selected MBIST 0 MBIST execution still ongoing 1 MBIST execution finished
23 MBESW8	MBESWx: On-Line End status (0 to 31) of the selected MBIST 0 MBIST execution still ongoing 1 MBIST execution finished
24 MBESW7	MBESWx: On-Line End status (0 to 31) of the selected MBIST 0 MBIST execution still ongoing 1 MBIST execution finished
25 MBESW6	MBESWx: On-Line End status (0 to 31) of the selected MBIST 0 MBIST execution still ongoing 1 MBIST execution finished
26 MBESW5	MBESWx: On-Line End status (0 to 31) of the selected MBIST 0 MBIST execution still ongoing 1 MBIST execution finished
27 MBESW4	MBESWx: On-Line End status (0 to 31) of the selected MBIST

Table continues on the next page...

STCU2_MBESW field descriptions (continued)

Field	Description
	0 MBIST execution still ongoing 1 MBIST execution finished
28 MBESW3	MBESWx: On-Line End status (0 to 31) of the selected MBIST 0 MBIST execution still ongoing 1 MBIST execution finished
29 MBESW2	MBESWx: On-Line End status (0 to 31) of the selected MBIST 0 MBIST execution still ongoing 1 MBIST execution finished
30 MBESW1	MBESWx: On-Line End status (0 to 31) of the selected MBIST 0 MBIST execution still ongoing 1 MBIST execution finished
31 MBESW0	MBESWx: On-Line End status (0 to 31) of the selected MBIST 0 MBIST execution still ongoing 1 MBIST execution finished

63.9.26 STCU2 On-Line MBIST End Flag Medium Register (STCU2_MBEMSW)

The STCU2_MBEMSW register includes the End Flag related to the execution of the selected On-Line MBIST in the range NMCUT = 32-63.

The size of the register depends on the number of BISTed RAMs/ROMs.

The content of this register is initialized to its reset value as soon as STCU2_RUNSW[RUNSW] = 1.

Address: 0h base + 6Ch offset = 6Ch

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	MBESW63	MBESW62	MBESW61	MBESW60	MBESW59	MBESW58	MBESW57	MBESW56	MBESW55	MBESW54	MBESW53	MBESW52	MBESW51	MBESW50	MBESW49	MBESW48
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Register description

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	MBESW47	MBESW46	MBESW45	MBESW44	MBESW43	MBESW42	MBESW41	MBESW40	MBESW39	MBESW38	MBESW37	MBESW36	MBESW35	MBESW34	MBESW33	MBESW32
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

STCU2_MBEMSW field descriptions

Field	Description
0 MBESW63	On-Line End status (32 to 63) of the selected MBIST 0 MBIST execution still ongoing 1 MBIST execution finished
1 MBESW62	On-Line End status (32 to 63) of the selected MBIST 0 MBIST execution still ongoing 1 MBIST execution finished
2 MBESW61	On-Line End status (32 to 63) of the selected MBIST 0 MBIST execution still ongoing 1 MBIST execution finished
3 MBESW60	On-Line End status (32 to 63) of the selected MBIST 0 MBIST execution still ongoing 1 MBIST execution finished
4 MBESW59	On-Line End status (32 to 63) of the selected MBIST 0 MBIST execution still ongoing 1 MBIST execution finished
5 MBESW58	On-Line End status (32 to 63) of the selected MBIST 0 MBIST execution still ongoing 1 MBIST execution finished
6 MBESW57	On-Line End status (32 to 63) of the selected MBIST 0 MBIST execution still ongoing 1 MBIST execution finished
7 MBESW56	On-Line End status (32 to 63) of the selected MBIST 0 MBIST execution still ongoing 1 MBIST execution finished

Table continues on the next page...

STCU2_MBEMSW field descriptions (continued)

Field	Description
8 MBESW55	On-Line End status (32 to 63) of the selected MBIST 0 MBIST execution still ongoing 1 MBIST execution finished
9 MBESW54	On-Line End status (32 to 63) of the selected MBIST 0 MBIST execution still ongoing 1 MBIST execution finished
10 MBESW53	On-Line End status (32 to 63) of the selected MBIST 0 MBIST execution still ongoing 1 MBIST execution finished
11 MBESW52	On-Line End status (32 to 63) of the selected MBIST 0 MBIST execution still ongoing 1 MBIST execution finished
12 MBESW51	On-Line End status (32 to 63) of the selected MBIST 0 MBIST execution still ongoing 1 MBIST execution finished
13 MBESW50	On-Line End status (32 to 63) of the selected MBIST 0 MBIST execution still ongoing 1 MBIST execution finished
14 MBESW49	On-Line End status (32 to 63) of the selected MBIST 0 MBIST execution still ongoing 1 MBIST execution finished
15 MBESW48	On-Line End status (32 to 63) of the selected MBIST 0 MBIST execution still ongoing 1 MBIST execution finished
16 MBESW47	On-Line End status (32 to 63) of the selected MBIST 0 MBIST execution still ongoing 1 MBIST execution finished
17 MBESW46	On-Line End status (32 to 63) of the selected MBIST 0 MBIST execution still ongoing 1 MBIST execution finished
18 MBESW45	On-Line End status (32 to 63) of the selected MBIST 0 MBIST execution still ongoing 1 MBIST execution finished
19 MBESW44	On-Line End status (32 to 63) of the selected MBIST 0 MBIST execution still ongoing 1 MBIST execution finished

Table continues on the next page...

STCU2_MBEMSW field descriptions (continued)

Field	Description
20 MBESW43	On-Line End status (32 to 63) of the selected MBIST 0 MBIST execution still ongoing 1 MBIST execution finished
21 MBESW42	On-Line End status (32 to 63) of the selected MBIST 0 MBIST execution still ongoing 1 MBIST execution finished
22 MBESW41	On-Line End status (32 to 63) of the selected MBIST 0 MBIST execution still ongoing 1 MBIST execution finished
23 MBESW40	On-Line End status (32 to 63) of the selected MBIST 0 MBIST execution still ongoing 1 MBIST execution finished
24 MBESW39	On-Line End status (32 to 63) of the selected MBIST 0 MBIST execution still ongoing 1 MBIST execution finished
25 MBESW38	On-Line End status (32 to 63) of the selected MBIST 0 MBIST execution still ongoing 1 MBIST execution finished
26 MBESW37	On-Line End status (32 to 63) of the selected MBIST 0 MBIST execution still ongoing 1 MBIST execution finished
27 MBESW36	On-Line End status (32 to 63) of the selected MBIST 0 MBIST execution still ongoing 1 MBIST execution finished
28 MBESW35	On-Line End status (32 to 63) of the selected MBIST 0 MBIST execution still ongoing 1 MBIST execution finished
29 MBESW34	On-Line End status (32 to 63) of the selected MBIST 0 MBIST execution still ongoing 1 MBIST execution finished
30 MBESW33	On-Line End status (32 to 63) of the selected MBIST 0 MBIST execution still ongoing 1 MBIST execution finished
31 MBESW32	On-Line End status (32 to 63) of the selected MBIST 0 MBIST execution still ongoing 1 MBIST execution finished

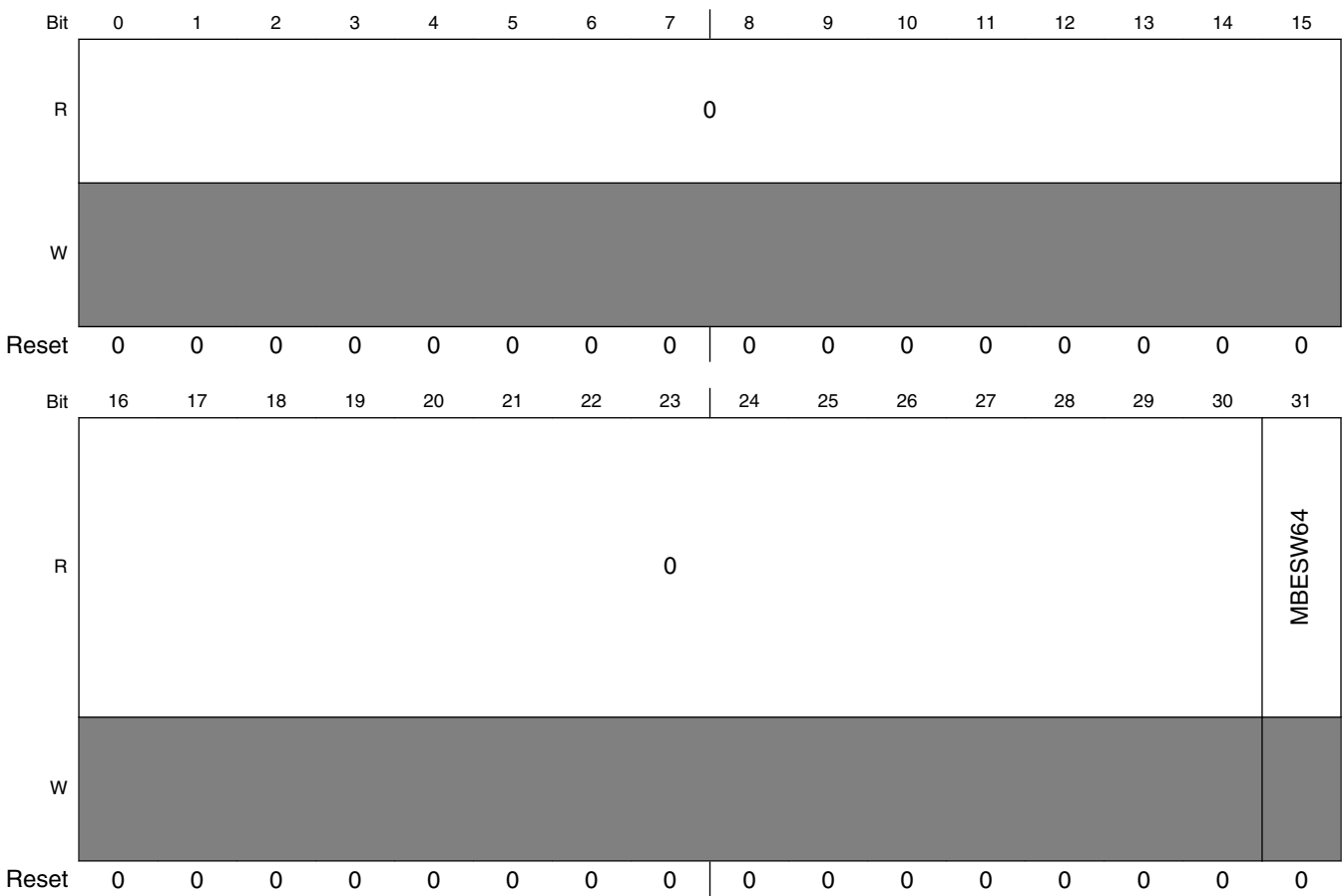
63.9.27 STCU2 On-Line MBIST End Flag High Register (STCU2_MBEHSW)

The STCU2_MBEHSW register includes the End Flag related to the execution of the selected On-Line MBIST in the range NMCUT = 64-95.

The size of the register depends on the number of BISTed RAMs/ROMs.

The content of this register is initialized to its reset value as soon as STCU2_RUNSW[RUNSW] = 1.

Address: 0h base + 70h offset = 70h



STCU2_MBEHSW field descriptions

Field	Description
0–30 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
31 MBESW64	MBESWx: On-Line End status (64 to 95) of the selected MBIST 0 MBIST execution still ongoing 1 MBIST execution finished

63.9.28 STCU2 MBIST Unrecoverable FM Low Register (STCU2_MBUFML)

NOTE

The reset value is chip-specific; see the chapter that describes how modules are configured and connected.

The STCU2_MBUFML register defines the fault mapping, in terms of Unrecoverable or Recoverable fault, of the MBIST in the range NMCUT = 0-31.

The size of the register depends on the number of BISTed RAMs/ROMs.

The R/W fields in this register are readable at any time. You can write to these fields when either of the following conditions is true:

- Off-line Self-Test phase is active
- On-line Self-Test phase is active and STCU2_CFG[WRP] = 0

Address: 0h base + 74h offset = 74h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	MBUFM31	MBUFM30	MBUFM29	MBUFM28	MBUFM27	MBUFM26	MBUFM25	MBUFM24	MBUFM23	MBUFM22	MBUFM21	MBUFM20	MBUFM19	MBUFM18	MBUFM17	MBUFM16
W	MBUFM31	MBUFM30	MBUFM29	MBUFM28	MBUFM27	MBUFM26	MBUFM25	MBUFM24	MBUFM23	MBUFM22	MBUFM21	MBUFM20	MBUFM19	MBUFM18	MBUFM17	MBUFM16
Reset	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	MBUFM15	MBUFM14	MBUFM13	MBUFM12	MBUFM11	MBUFM10	MBUFM9	MBUFM8	MBUFM7	MBUFM6	MBUFM5	MBUFM4	MBUFM3	MBUFM2	MBUFM1	MBUFM0
W	MBUFM15	MBUFM14	MBUFM13	MBUFM12	MBUFM11	MBUFM10	MBUFM9	MBUFM8	MBUFM7	MBUFM6	MBUFM5	MBUFM4	MBUFM3	MBUFM2	MBUFM1	MBUFM0
Reset	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*

* Notes:

- The reset value is chip-specific; see the chapter that describes how modules are configured and connected.x = Undefined at reset.

STCU2_MBUFML field descriptions

Field	Description
0 MBUFM31	MBIST Unrecoverable Fault Mapping 0 Recoverable Fault mapping 1 Unrecoverable Fault mapping
1 MBUFM30	MBIST Unrecoverable Fault Mapping

Table continues on the next page...

STCU2_MBUFML field descriptions (continued)

Field	Description
	0 Recoverable Fault mapping 1 Unrecoverable Fault mapping
2 MBUFM29	MBIST Unrecoverable Fault Mapping 0 Recoverable Fault mapping 1 Unrecoverable Fault mapping
3 MBUFM28	MBIST Unrecoverable Fault Mapping 0 Recoverable Fault mapping 1 Unrecoverable Fault mapping
4 MBUFM27	MBIST Unrecoverable Fault Mapping 0 Recoverable Fault mapping 1 Unrecoverable Fault mapping
5 MBUFM26	MBIST Unrecoverable Fault Mapping 0 Recoverable Fault mapping 1 Unrecoverable Fault mapping
6 MBUFM25	MBIST Unrecoverable Fault Mapping 0 Recoverable Fault mapping 1 Unrecoverable Fault mapping
7 MBUFM24	MBIST Unrecoverable Fault Mapping 0 Recoverable Fault mapping 1 Unrecoverable Fault mapping
8 MBUFM23	MBIST Unrecoverable Fault Mapping 0 Recoverable Fault mapping 1 Unrecoverable Fault mapping
9 MBUFM22	MBIST Unrecoverable Fault Mapping 0 Recoverable Fault mapping 1 Unrecoverable Fault mapping
10 MBUFM21	MBIST Unrecoverable Fault Mapping 0 Recoverable Fault mapping 1 Unrecoverable Fault mapping
11 MBUFM20	MBIST Unrecoverable Fault Mapping 0 Recoverable Fault mapping 1 Unrecoverable Fault mapping
12 MBUFM19	MBIST Unrecoverable Fault Mapping 0 Recoverable Fault mapping 1 Unrecoverable Fault mapping
13 MBUFM18	MBIST Unrecoverable Fault Mapping

Table continues on the next page...

STCU2_MBUFML field descriptions (continued)

Field	Description
	0 Recoverable Fault mapping 1 Unrecoverable Fault mapping
14 MBUFM17	MBIST Unrecoverable Fault Mapping 0 Recoverable Fault mapping 1 Unrecoverable Fault mapping
15 MBUFM16	MBIST Unrecoverable Fault Mapping 0 Recoverable Fault mapping 1 Unrecoverable Fault mapping
16 MBUFM15	MBIST Unrecoverable Fault Mapping 0 Recoverable Fault mapping 1 Unrecoverable Fault mapping
17 MBUFM14	MBIST Unrecoverable Fault Mapping 0 Recoverable Fault mapping 1 Unrecoverable Fault mapping
18 MBUFM13	MBIST Unrecoverable Fault Mapping 0 Recoverable Fault mapping 1 Unrecoverable Fault mapping
19 MBUFM12	MBIST Unrecoverable Fault Mapping 0 Recoverable Fault mapping 1 Unrecoverable Fault mapping
20 MBUFM11	MBIST Unrecoverable Fault Mapping 0 Recoverable Fault mapping 1 Unrecoverable Fault mapping
21 MBUFM10	MBIST Unrecoverable Fault Mapping 0 Recoverable Fault mapping 1 Unrecoverable Fault mapping
22 MBUFM9	MBIST Unrecoverable Fault Mapping 0 Recoverable Fault mapping 1 Unrecoverable Fault mapping
23 MBUFM8	MBIST Unrecoverable Fault Mapping 0 Recoverable Fault mapping 1 Unrecoverable Fault mapping
24 MBUFM7	MBIST Unrecoverable Fault Mapping 0 Recoverable Fault mapping 1 Unrecoverable Fault mapping
25 MBUFM6	MBIST Unrecoverable Fault Mapping

Table continues on the next page...

STCU2_MBUFML field descriptions (continued)

Field	Description
	0 Recoverable Fault mapping 1 Unrecoverable Fault mapping
26 MBUFM5	MBIST Unrecoverable Fault Mapping 0 Recoverable Fault mapping 1 Unrecoverable Fault mapping
27 MBUFM4	MBIST Unrecoverable Fault Mapping 0 Recoverable Fault mapping 1 Unrecoverable Fault mapping
28 MBUFM3	MBIST Unrecoverable Fault Mapping 0 Recoverable Fault mapping 1 Unrecoverable Fault mapping
29 MBUFM2	MBIST Unrecoverable Fault Mapping 0 Recoverable Fault mapping 1 Unrecoverable Fault mapping
30 MBUFM1	MBIST Unrecoverable Fault Mapping 0 Recoverable Fault mapping 1 Unrecoverable Fault mapping
31 MBUFM0	MBIST Unrecoverable Fault Mapping 0 Recoverable Fault mapping 1 Unrecoverable Fault mapping

63.9.29 STCU2 MBIST Unrecoverable FM Medium Register (STCU2_MBUFMM)**NOTE**

The reset value is chip-specific; see the chapter that describes how modules are configured and connected.

The STCU2_MBUFMM register defines the fault mapping, in terms of Unrecoverable or Recoverable fault, of the MBIST in the range NMCUT = 32-63.

The size of the register depends on the number of BISTed RAMs/ROMs.

The R/W fields in this register are readable at any time. You can write to these fields when either of the following conditions is true:

- Off-line Self-Test phase is active
- On-line Self-Test phase is active and STCU2_CFG[WRP] = 0

Register description

Address: 0h base + 78h offset = 78h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	MBUFM63	MBUFM62	MBUFM61	MBUFM60	MBUFM59	MBUFM58	MBUFM57	MBUFM56	MBUFM55	MBUFM54	MBUFM53	MBUFM52	MBUFM51	MBUFM50	MBUFM49	MBUFM48
W																
Reset	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	MBUFM47	MBUFM46	MBUFM45	MBUFM44	MBUFM43	MBUFM42	MBUFM41	MBUFM40	MBUFM39	MBUFM38	MBUFM37	MBUFM36	MBUFM35	MBUFM34	MBUFM33	MBUFM32
W																
Reset	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*

* Notes:

- The reset value is chip-specific; see the chapter that describes how modules are configured and connected.x = Undefined at reset.

STCU2_MBUFMM field descriptions

Field	Description
0 MBUFM63	MBIST Unrecoverable Fault Mapping 0 Recoverable Fault mapping 1 Unrecoverable Fault mapping
1 MBUFM62	MBIST Unrecoverable Fault Mapping 0 Recoverable Fault mapping 1 Unrecoverable Fault mapping
2 MBUFM61	MBIST Unrecoverable Fault Mapping 0 Recoverable Fault mapping 1 Unrecoverable Fault mapping
3 MBUFM60	MBIST Unrecoverable Fault Mapping 0 Recoverable Fault mapping 1 Unrecoverable Fault mapping
4 MBUFM59	MBIST Unrecoverable Fault Mapping 0 Recoverable Fault mapping 1 Unrecoverable Fault mapping
5 MBUFM58	MBIST Unrecoverable Fault Mapping 0 Recoverable Fault mapping 1 Unrecoverable Fault mapping
6 MBUFM57	MBIST Unrecoverable Fault Mapping 0 Recoverable Fault mapping 1 Unrecoverable Fault mapping

Table continues on the next page...

STCU2_MBUFMM field descriptions (continued)

Field	Description
7 MBUFM56	MBIST Unrecoverable Fault Mapping 0 Recoverable Fault mapping 1 Unrecoverable Fault mapping
8 MBUFM55	MBIST Unrecoverable Fault Mapping 0 Recoverable Fault mapping 1 Unrecoverable Fault mapping
9 MBUFM54	MBIST Unrecoverable Fault Mapping 0 Recoverable Fault mapping 1 Unrecoverable Fault mapping
10 MBUFM53	MBIST Unrecoverable Fault Mapping 0 Recoverable Fault mapping 1 Unrecoverable Fault mapping
11 MBUFM52	MBIST Unrecoverable Fault Mapping 0 Recoverable Fault mapping 1 Unrecoverable Fault mapping
12 MBUFM51	MBIST Unrecoverable Fault Mapping 0 Recoverable Fault mapping 1 Unrecoverable Fault mapping
13 MBUFM50	MBIST Unrecoverable Fault Mapping 0 Recoverable Fault mapping 1 Unrecoverable Fault mapping
14 MBUFM49	MBIST Unrecoverable Fault Mapping 0 Recoverable Fault mapping 1 Unrecoverable Fault mapping
15 MBUFM48	MBIST Unrecoverable Fault Mapping 0 Recoverable Fault mapping 1 Unrecoverable Fault mapping
16 MBUFM47	MBIST Unrecoverable Fault Mapping 0 Recoverable Fault mapping 1 Unrecoverable Fault mapping
17 MBUFM46	MBIST Unrecoverable Fault Mapping 0 Recoverable Fault mapping 1 Unrecoverable Fault mapping
18 MBUFM45	MBIST Unrecoverable Fault Mapping 0 Recoverable Fault mapping 1 Unrecoverable Fault mapping

Table continues on the next page...

STCU2_MBUFMM field descriptions (continued)

Field	Description
19 MBUFM44	MBIST Unrecoverable Fault Mapping 0 Recoverable Fault mapping 1 Unrecoverable Fault mapping
20 MBUFM43	MBIST Unrecoverable Fault Mapping 0 Recoverable Fault mapping 1 Unrecoverable Fault mapping
21 MBUFM42	MBIST Unrecoverable Fault Mapping 0 Recoverable Fault mapping 1 Unrecoverable Fault mapping
22 MBUFM41	MBIST Unrecoverable Fault Mapping 0 Recoverable Fault mapping 1 Unrecoverable Fault mapping
23 MBUFM40	MBIST Unrecoverable Fault Mapping 0 Recoverable Fault mapping 1 Unrecoverable Fault mapping
24 MBUFM39	MBIST Unrecoverable Fault Mapping 0 Recoverable Fault mapping 1 Unrecoverable Fault mapping
25 MBUFM38	MBIST Unrecoverable Fault Mapping 0 Recoverable Fault mapping 1 Unrecoverable Fault mapping
26 MBUFM37	MBIST Unrecoverable Fault Mapping 0 Recoverable Fault mapping 1 Unrecoverable Fault mapping
27 MBUFM36	MBIST Unrecoverable Fault Mapping 0 Recoverable Fault mapping 1 Unrecoverable Fault mapping
28 MBUFM35	MBIST Unrecoverable Fault Mapping 0 Recoverable Fault mapping 1 Unrecoverable Fault mapping
29 MBUFM34	MBIST Unrecoverable Fault Mapping 0 Recoverable Fault mapping 1 Unrecoverable Fault mapping
30 MBUFM33	MBIST Unrecoverable Fault Mapping 0 Recoverable Fault mapping 1 Unrecoverable Fault mapping

Table continues on the next page...

STCU2_MBUFMM field descriptions (continued)

Field	Description
31 MBUFM32	MBIST Unrecoverable Fault Mapping 0 Recoverable Fault mapping 1 Unrecoverable Fault mapping

63.9.30 STCU2 MBIST Unrecoverable FM High Register (STCU2_MBUFMMH)**NOTE**

The reset value is chip-specific; see the chapter that describes how modules are configured and connected.

The STCU2_MBUFMMH register defines the fault mapping, in terms of Unrecoverable or Recoverable fault, of the MBIST in the range NMCUT = 64-95.

The size of the register depends on the number of BISTed RAMs/ROMs.

The R/W fields in this register are readable at any time. You can write to these fields when either of the following conditions is true:

- Off-line Self-Test phase is active
- On-line Self-Test phase is active and STCU2_CFG[WRP] = 0

Address: 0h base + 7Ch offset = 7Ch

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	
R	0																
W																	
Reset	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	
Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	
R	0															MBUFM64	
W																	
Reset	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*		

* Notes:

- The reset value is chip-specific; see the chapter that describes how modules are configured and connected. x = Undefined at reset.

STCU2_MBUFMH field descriptions

Field	Description
0–30 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
31 MBUFM64	MBIST Unrecoverable Fault Mapping 0 Recoverable Fault mapping 1 Unrecoverable Fault mapping

63.9.31 STCU2 LBIST Control (STCU2_LB_CTRLn)**NOTE**

The reset value is chip-specific; see the chapter that describes how modules are configured and connected.

The STCU2_LB_CTRL register defines the control setting of each LBIST controller.

The R/W fields in this register are readable at any time. You can write to these fields when either of the following conditions is true:

- Offline self test phase is active
- Online self test phase is active and STCU2_CFG[WRP] = 0

Address: 0h base + 100h offset + (64d × i), where i=0d to 5d

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	CSM							PTR			0		Reserved	Reserved	SHS	
W																
Reset	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*
Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	SCEN_OFF				SCEN_ON				0				PFT	CWS		
W																
Reset	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*

* Notes:

- The reset value is chip-specific; see the chapter that describes how modules are configured and connected.x = Undefined at reset.

STCU2_LB_CTRLn field descriptions

Field	Description
0 CSM	Concurrent/sequential mode

Table continues on the next page...

STCU2_LB_CTRLn field descriptions (continued)

Field	Description
	The next LBIST is scheduled concurrently to the current one if the CSM bit is set to 1; otherwise, it is scheduled sequentially to the completion of the current LBIST execution. 0 Sequential mode 1 Concurrent mode
1–7 PTR	Next LBIST pointer PTR defines the logical pointer to the next LBIST. In case of NIL pointer the CSM bit has to be set Sequential (0) to define this is the end of the list. The self testing procedure is stopped when the current LBIST has been completed. LBIST POINTERS 0000b - pointer to LBIST partition 0 0001b - pointer to LBIST partition 1 0010b - pointer to LBIST partition 2 0011b - pointer to LBIST partition 3 0100b - pointer to LBIST partition 4 0101b - pointer to LBIST partition 5 ... 0x7F - pointer to NIL, end of LBIST execution Values not defined in this list will result in an error being set in the STCU2_ERR register.
8–10 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
11 Reserved	This field is reserved. NOTE: This reserved field is writeable but do not write any value to it other than its reset value
12 Reserved	This field is reserved. NOTE: This reserved field is writeable but do not write any value to it other than its reset value
13–15 SHS	Shift speed SHS defines the shift speed 000 Shift at full rate of STCU2 core clock 001 Shift at 1/2 rate of STCU2 core clock 010 Shift at 1/3 rate of STCU2 core clock 011 Shift at 1/4 rate of STCU2 core clock 100 Shift at 1/5 rate of STCU2 core clock 101 Shift at 1/6 rate of STCU2 core clock 110 Shift at 1/7 rate of STCU2 core clock 111 Shift at 1/8 rate of STCU2 core clock
16–19 SCEN_OFF	Scan enable OFF SCEN_OFF defines the number of clock cycles OFF following the falling transition on the SCEN NOTE: Scen_off must be programmed to a value >=1 0000 0 delay cycles 0001 1 delay cycle

Table continues on the next page...

STCU2_LB_CTRLn field descriptions (continued)

Field	Description
	0010 2 delay cycles 0011 3 delay cycles 0100 4 delay cycles 0101 5 delay cycles 0110 6 delay cycles 0111 7 delay cycles 1000 8 delay cycles 1001 9 delay cycles 1010 10 delay cycles 1011 11 delay cycles 1100 12 delay cycles 1101 13 delay cycles 1110 14 delay cycles 1111 15 delay cycles
20–23 SCEN_ON	Scan enable ON SCEN_ON defines the number of clock cycles OFF following the rising transition on the SCEN NOTE: Scen_on delay register value must be programmed to a value ≥ 1 0000 0 delay cycles 0001 1 delay cycle 0010 2 delay cycles 0011 3 delay cycles 0100 4 delay cycles 0101 5 delay cycles 0110 6 delay cycles 0111 7 delay cycles 1000 8 delay cycles 1001 9 delay cycles 1010 10 delay cycles 1011 11 delay cycles 1100 12 delay cycles 1101 13 delay cycles 1110 14 delay cycles 1111 15 delay cycles
24–27 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
28 PFT	Past Flush Test The LBIST controller, by default, applies 32 Flush Test patterns. This bit allows to skip/apply the Flush test pattern sequence. 0 Apply Flush Test Patterns 1 Skip Flush Test Patterns
29–31 CWS	Capture window size CWS defines the capture window size. 000 illegal

Table continues on the next page...

STCU2_LB_CTRLn field descriptions (continued)

Field	Description
001	controller waits 1 shift cycle for capture to finish
010	controller waits 2 shift cycles for capture to finish
011	controller waits 3 shift cycles for capture to finish
100	controller waits 4 shift cycles for capture to finish
101	controller waits 5 shift cycles for capture to finish
110	controller waits 6 shift cycles for capture to finish
111	controller waits 7 shift cycles for capture to finish

63.9.32 STCU2 LBIST PC Stop Register (STCU2_LB_PCSn)**NOTE**

The reset value is chip-specific; see the chapter that describes how modules are configured and connected.

The STCU2_LB_PCS register defines the pattern counter stop of each LBIST controller.

The R/W fields in this register are readable at any time. You can write to these fields when either of the following conditions is true:

- Off-line Self-Test phase is active
- On-line Self-Test phase is active and STCU2_CFG[WRP] = 0

Address: 0h base + 104h offset + (64d × i), where i=0d to 5d

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0						PCS																									
W																																
Reset	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*

* Notes:

- The reset value is chip-specific; see the chapter that describes how modules are configured and connected. x = Undefined at reset.

STCU2_LB_PCSn field descriptions

Field	Description
0–5 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
6–31 PCS	Pattern counter stop PCS defines the pattern counter stop value.

63.9.33 STCU2 Off-Line LBIST MISR Expected Low Register (STCU2_LB_MISREL_n)

NOTE

The reset value is chip-specific; see the chapter that describes how modules are configured and connected.

The STCU2_LB_MISREL register defines the LSB part (31-0) of the Expected MISR of the Off-Line LBIST controller.

The size of the register depends on the number of MISR bits of the related LBIST.

The R/W fields in this register are readable at any time. However, you can write to these fields only when Off-line Self-Test phase is still active.

Address: 0h base + 110h offset + (64d × i), where i=0d to 5d

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	MISREx																															
W																																
Reset	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*

- * Notes:
- The reset value is chip-specific; see the chapter that describes how modules are configured and connected.x = Undefined at reset.

STCU2_LB_MISREL_n field descriptions

Field	Description
0–31 MISREx	Off-Line MISR Expected low Bits This field defines the low part (31..0) of the Expected MISR.

63.9.34 STCU2 Off-Line LBIST MISR Expected High Register (STCU2_LB_MISREH_n)

NOTE

The reset value is chip-specific; see the chapter that describes how modules are configured and connected.

The STCU2_LB_MISREH register defines the MSB part (63-32) of the Expected MISR of the Off-Line LBIST controller.

The size of the register depends on the number of MISR bits of the related LBIST.

The R/W fields in this register are readable at any time. However, you can write to these fields only when Off-line Self-Test phase is still active.

Address: 0h base + 114h offset + (64d × i), where i=0d to 5d

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	MISREx																															
W																																
Reset	X*	X*	X*	X*	X*	X*	X*	X*	X*	X*	X*	X*	X*	X*	X*	X*	X*	X*	X*	X*	X*	X*	X*	X*	X*	X*	X*	X*	X*	X*	X*	

* Notes:

- The reset value is chip-specific; see the chapter that describes how modules are configured and connected.x = Undefined at reset.

STCU2_LB_MISREHn field descriptions

Field	Description
0–31 MISREx	Off-Line MISR Expected high Bits This field defines the high part (63-32) of the Expected MISR.

63.9.35 STCU2 Off-Line LBIST MISR Read Low Register (STCU2_LB_MISRRLn)

NOTE

The reset value is chip-specific; see the chapter that describes how modules are configured and connected.

The STCU2_LB_MISRRL register reports the LSB part (31-0) of the MISR obtained at the end of the Off-Line LBIST controller execution.

The size of the register depends on the number of MISR bits of the related LBIST.

Address: 0h base + 118h offset + (64d × i), where i=0d to 5d

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	MISRRx																															
W																																
Reset	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	

* Notes:

- The reset value is chip-specific; see the chapter that describes how modules are configured and connected.x = Undefined at reset.

STCU2_LB_MISRRLn field descriptions

Field	Description
0–31 MISRRLx	Off-Line MISR Read Low Bits This field is equivalent to the Low Bits (31-0) of the MISR obtained at the end of the Off-Line LBIST execution

63.9.36 STCU2 Off-Line LBIST MISR Read High Register (STCU2_LB_MISRRHn)

NOTE

The reset value is chip-specific; see the chapter that describes how modules are configured and connected.

The STCU2_LB_MISRRH register report the MSB part (63-32) of the MISR obtained at the end of each Off-Line LBIST controller execution.

The size of the register depends on the number of MISR bits of the related LBIST.

Address: 0h base + 11Ch offset + (64d × i), where i=0d to 5d

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	MISRRx																															
W																																
Reset	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*

- * Notes:
- The reset value is chip-specific; see the chapter that describes how modules are configured and connected.x = Undefined at reset.

STCU2_LB_MISRRHn field descriptions

Field	Description
0–31 MISRRx	Off-Line MISR Read High Bits This field is equivalent to the High Bits (63-32) of the MISR obtained at the end of the Off-Line LBIST execution

63.9.37 STCU2 On-Line LBIST MISR Expected Low Register (STCU2_LB_MISRELSWn)

NOTE

The reset value is chip-specific; see the chapter that describes how modules are configured and connected.

The STCU2_LB_MISRELSW register defines the LSB part (31-0) of the Expected MISR of the On-Line LBIST controller.

The size of the register depends on the number of MISR bits of the related LBIST.

The R/W fields in this register are readable at any time. You can write to these fields when either of the following conditions is true:

- Off-line Self-Test phase is active
- On-line Self-Test phase is active and STCU2_CFG[WRP] = 0

Address: 0h base + 120h offset + (64d × i), where i=0d to 5d

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	MISRESWx																															
W																																
Reset	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	

* Notes:

- The reset value is chip-specific; see the chapter that describes how modules are configured and connected.x = Undefined at reset.

STCU2_LB_MISRELSW_n field descriptions

Field	Description
0–31 MISRESWx	On-Line MISR Expected low Bits This field defines the low part (31..0) of the Expected MISR.

63.9.38 STCU2 On-Line LBIST MISR Expected High Register (STCU2_LB_MISREHSW_n)

NOTE

The reset value is chip-specific; see the chapter that describes how modules are configured and connected.

The STCU2_LB_MISREHSW register defines the MSB part (63-32) of the Expected MISR of the On-Line LBIST controller.

The size of the register depends on the number of MISR bits of the related LBIST.

The R/W fields in this register are readable at any time. You can write to these fields when either of the following conditions is true:

- Off-line Self-Test phase is active
- On-line Self-Test phase is active and STCU2_CFG[WRP] = 0

Address: 0h base + 124h offset + (64d × i), where i=0d to 5d

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	MISRESW _x																															
W																																
Reset	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*

* Notes:

- The reset value is chip-specific; see the chapter that describes how modules are configured and connected.x = Undefined at reset.

STCU2_LB_MISREHSW_n field descriptions

Field	Description
0–31 MISRESW _x	On-Line MISR Expected high Bits This field defines the high part (63-32) of the Expected MISR.

63.9.39 STCU2 On-Line LBIST MISR Read Low Register (STCU2_LB_MISRRLSW_n)**NOTE**

The reset value is chip-specific; see the chapter that describes how modules are configured and connected.

The STCU2_LB_MISRRLSW register report the LSB part (31-0) of the MISR obtained at the end of the On-Line LBIST controller execution.

The size of the register depends on the number of MISR bits of the related LBIST.

The content of this register is initialized to its reset value as soon as STCU2_RUNSW[RUNSW] = 1.

Address: 0h base + 128h offset + (64d × i), where i=0d to 5d

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	MISRRSW _x																															
W																																
Reset	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*

* Notes:

- The reset value is chip-specific; see the chapter that describes how modules are configured and connected. x = Undefined at reset.

STCU2_LB_MISRRLSW_n field descriptions

Field	Description
0–31 MISRRSW _x	On-Line MISR Read Low Bin This field is equivalent to the Low Bits (31-0) of the MISR obtained at the end of the Off-Line LBIST execution.

63.9.40 STCU2 On-Line LBIST MISR Read High Register (STCU2_LB_MISRRHSW_n)**NOTE**

The reset value is chip-specific; see the chapter that describes how modules are configured and connected.

The STCU2_LB_MISRRHSW register report the MSB part (63-32) of the MISR obtained at the end of each On-Line LBIST controller execution.

The size of the register depends on the number of MISR bits of the related LBIST.

The content of this register is initialized to its reset value as soon as STCU2_RUNSW[RUNSW] = 1.

Address: 0h base + 12Ch offset + (64d × i), where i=0d to 5d

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	MISRRSWx																															
W																																
Reset	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	

* Notes:

- The reset value is chip-specific; see the chapter that describes how modules are configured and connected.x = Undefined at reset.

STCU2_LB_MISRRHSW_n field descriptions

Field	Description
0–31 MISRRSWx	On-Line MISR Read High Bits This field is equivalent to the Low Bits (63-32) of the MISR obtained at the end of the Off-Line LBIST execution.

63.9.41 STCU2 MBIST Control Register (STCU2_MB_CTRL_n)

NOTE

The reset value is chip-specific; see the chapter that describes how modules are configured and connected.

The STCU2_MB_CTRL register defines the control setting of MBIST controller.

The R/W fields in this register are readable at any time. You can write to these fields when either of the following conditions is true:

- Off-line Self-Test phase is active
- On-line Self-Test phase is active and STCU2_CFG[WRP] = 0

The offset of this register is a function of NMCUT .

Address: 0h base + 600h offset + (4d × i), where i=0d to 64d

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	
R	CSM		PTR						0								
W																	
Reset	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	

Use cases and limitations

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0															
W																
Reset	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*

* Notes:

- The reset value is chip-specific; see the chapter that describes how modules are configured and connected.x = Undefined at reset.

STCU2_MB_CTRLn field descriptions

Field	Description
0 CSM	Concurrent/sequential mode 0 Sequential mode 1 Concurrent mode
1–7 PTR	Next LBIST or MBIST pointer PTR defines the logical pointer to the next LBIST or MBIST to be scheduled. The next LBIST or MBIST is scheduled concurrently to the current one if the CSM bit is set to 1; otherwise it is scheduled sequentially to the completion of the current MBIST execution. In case of NIL pointer the CSM bit must be set Sequential (0) to define this is the end of the list. The self-testing procedure is stopped when the current MBIST has been completed. 0h to (LBIST -1): pointer to LBIST 10h to (10h + MBIST -1b): pointer to MBIST 7Fh: pointer to NIL. No BIST execution. others: invalid pointer => an error is set into the STCU2_ERR register.
8–31 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

63.10 Use cases and limitations

The STCU2 module is designed to give high flexibility during the on/off-line test. In the following are reported the programming operations that have to be performed to correctly use the functionality of this module considering the three typical usage cases:

- the Off-line Self Test Sequence after a reset trigger event
- the bypass mode of the Self Test Sequence after a reset trigger event
- the On-line Self-test Sequence

63.10.1 Offline self-test sequence

This is the typical mode of using the STCU2 module after reset trigger event is applied to STCU2. The SSCM DCF bus is used to retrieve the STCU2 schedule and L/MBIST execution parameters stored into the NVM memory. The target is to cover the amount of physical defects in the digital logic and in the System RAMs/ROMs according to the System specifications. The sequence is the following:

- Unlock the Off-Line STCU2 access writing the Key1/Key2 sequence into the STCU2_SKC register
- Program the LBIST fault reaction conditions (UF/RF) setting the STCU2_LBUFM register
- Program the MBIST fault reaction conditions (UF/RF) setting, depending on the number of memories, the STCU2_MBUFML/M/H registers
- Program the STCU2 internal fault reaction conditions (UF/RF) setting the STCU2_ERR_FM register
- Program the STCU2_LB_CTRL, STCU2_LB_PCS, STCU2_MISREL/H registers of each LBIST to be executed.
- Program the STCU2_MB_CTRL registers of each NMCUT to be executed
- Program the STCU2_WDG register to define the time budget assigned for the L/MBIST execution
- Program the STCU2_CFG register to define: the core and LBIST/MBIST TCK clock prescaling factor setting the CLK_CFG bits, the LB_DELAY field to delay the LBIST start when many LBIST are run concurrently to mitigate the potential transient power issue during the start-up phase, the PMOSEN or MBU in order to define the algorithm to be used during MBIST run, and finally set the pointer to the first LBIST/NMCUT to be executed
- In case the PLL usage has been enabled, program the STCU2_PLL_CFG value in order to set the required system frequency
- Program the RUN bit into the STCU2_RUN register to enable the Off-Line Self-Test execution and the L/MBPLEN bits to enable the PLL during L/MBIST execution according to STCU2_PLL_CFG register content
- Start the Off-Line Self-Test and, depending on PLL management and L/MBIST execution order, when PLL LOCK is high.
- Wait the WDGEOC run time execution specified into the STCU2_WDG register.
- The STCU2 switches off the core clock at the end of the Self-Test run and releases the PLL control in case this feature has been enabled.
- The STCU2 resets the STCU2_RUN[RUN] bit and the device exits from boot sequence. In case there are failures, the STCU2 flags these failures toward the FCCU.
- The software application reads the STCU2_LBS flag register to check the failed LBIST when UF or RF conditions is/are detected.
- The software application reads the STCU2_LBE flag register to check the LBIST still on-going when UF or RF is/are detected.

- The software application reads the STCU2_MBSL/M/H flag registers, depending on the number of memories, to check the failed RAMs/ROMs Memory BIST when UF or RF is/are detected.
- The software application reads the STCU2_MBEL/M/H flag registers, depending on the number of memories, to check if the RAMs/ROMs Memory BIST is still on-going when UF or RF is/are detected.
- The software application reads the bits INV, ENGE, WDTO, LOCKE of the STCU2_ERR_STAT register to check whether there has been an internal STCU2 engine/parameters failure when UF or RF is/are detected.
- The software application reads for each device's LBIST, the STCU2_LBMISREL/H and STCU2_LBIST_MISRRL/H registers to check the coherence with the STCU2_LBS bits.

NOTE

The software application reads operations on the STCU2 and LBIST MISR registers after off-line self test execution has been successfully performed are just an additional reliability layer added to improve the already implemented Auto Self-Test feature included into STCU2.

63.10.2 Online self test sequence

The online self test sequence is the most commonly used STCU2 application mode. The IPS interface is used to program the STCU2 registers and to schedule LBIST and MBIST execution. Since online LBIST and MBIST operations clean-up the content of Digital Logic (LBIST) or RAM (MBIST) tested, activation of this mode must be approached with caution. The suggested sequence is the following:

1. Write the Key1/Key2 sequence to the SKC register to unlock online STCU2 access.
2. Ensure CFG[WRP]=0 to enable IPS bus access to the online self test registers.
3. If required, program the following registers to select the LBIST, MBIST and STCU2 internal fault reaction: LBUFM, MBUFML/M/H and ERR_FM.

NOTE

The LBIST, MBIST and STCU2 internal fault reaction should have been already programmed during the offline self test sequence and so there should be no need to program again. However, in case they have not been

programmed or there is the need of changing them for any reason, they can be overwritten.

4. Select the reset management options for online LBIST execution by programming the LBRMSW register.
5. The LB_CTRL and LB_PCS registers should have been already programmed during offline self test sequence and so they should not need to be programmed again. However, if they have not been programmed or there is the need of changing them for any reason, they can be overwritten. The registers LB_MISRELSW/HSW have to be programmed according to the expected signature.
6. Overwrite/Program the MB_CTRL registers of each NMCUT to be executed.
7. Overwrite/Program the WDG register to define the time budget assigned for the L/MBIST execution
8. Overwrite/Program the CFG register to define: the core and L/MBIST TCK clock prescaling factor setting the CLK_CFG bits, the LB_DELAY field to delay the LBIST start when many LBIST are run concurrently to mitigate the potential transient power issue during the start-up phase, the PMOSEN or MBU in order to define the algorithm to be used during MBIST run, and finally set the pointer to the first LBIST/NMCUT to be executed.
9. Program the RUNSW[RUNSW] field to enable online self test execution; the L/MBSWPLEN bits to enable the PLL lock signal monitor during L/MBIST execution and the MBIE to enable the related Interrupt requests.
10. Start the online self test and depending on PLL management and L/MBIST execution order, also when PLL LOCK is high.
11. Wait the WDGEOC run time execution specified into the WDG register
12. In case the MBIE have been enabled, an interrupt is generated at the end of all the MBIST execution. The software application has to manage these interrupt signals and clean-up the related flag (MBIFLG) into the INT_FLG register.
13. The STCU2 switch-off the core clock at the end of the self test run
14. The functional reset activated is managed by the LBRMSW register (See step 4).
15. In case, at least one of the selected LBIST enables the Global functional reset, the STCU2 requests a functional reset which restarts the system while, when all the selected LBIST enables its own dedicated functional reset, only these local reset lines are applied and only the related LBIST partitions are cleaned-up.
16. The STCU2 resets the RUNSW[RUNSW] field. In both the reset cases and in case of fault/s, the STCU2 flags the failure toward the FCCU.
17. The software application reads the LBSSW flag register to check the failed LBIST when UF or RF conditions is/are detected.
18. The software application reads the LBESW flag register to check if the LBIST is still on-going when UF or RF is/are detected.

19. The software application reads the MBSLSW/MSW/HSW flag registers, depending on the number of memories, to check the failed RAMs/ROMs Memory BIST when UF or RF is/are detected.
20. The software application reads the MBELSW/MSW/HSW flag registers, depending on the number of memories, to check if the RAMs/ROMs Memory BIST is still on-going when UF or RF is/are detected.
21. The software application reads the bit INVPSW, ENGESW, WDTOSW, LOCKESW of the ERR_STAT register to check whether there has been an internal STCU2 engine/parameters failure when UF or RF is/are detected.
22. The software application reads for each device's LBIST, the LBMISRELSW/HSW and LBIST_MISRRLSW/HSW registers to check the coherence with the LBS bits

NOTE

1. A reset is necessary after running any kind of online self test.
2. The software application read operations on the STCU2 and LBIST MISR registers after successful online self test execution are an additional reliability layer to improve the auto-self-test feature included in the STCU2.
3. Flash memory must be in the idle state when BIST execution is started. Program and erase functionality must not be enabled when entering BIST execution.

63.10.3 Bypass USER Mode

In this case, the USER application parameters stored in FLASH and read by SSCM are written in order to skip the Self-Test procedure after a reset trigger event is applied. It might be useful in case the device is not configured for applications requiring improved reliability. The sequence is the following:

1. Unlock the STCU2 access writing the Off-Line Key1/Key2 sequence into the STCU2_SKC register
2. Set the BYP bit in the STCU2_RUN register

After BYP bit is set, the core clock is switched off, the Self-Test sequence is not applied and the system can wake up and start the user application.

63.10.4 Design implementation information

The following list reports the limitations related to the current implementation:

- Every NMCUT BIST can be run once during the online or offline self-test procedure.
- It is not possible to run LBIST and MBIST concurrently.
- Because software has full control of the system while applications are running during the online self-test, STCU2 is not allowed to take control of the PLL but can just monitor the lock signal to flag wrong unlock events.

Chapter 64

Cyclic Redundancy Check (CRC) Unit

64.1 Chip-specific CRC information

This chip has one instance of the CRC module. It supports three contexts: $N = 3$.

64.2 Introduction

The Cyclic Redundancy Check (CRC) computing unit is dedicated to the computation of CRC, off-loading the CPU. Each context has a separate CRC computation engine in order to allow the concurrent computation of the CRC of multiple data streams. The CRC computation is performed at speed without wait-state insertion. Bit-swap and bit-inversion operations can be applied on the final CRC signature. Each context can be configured with one of three hard-wired polynomials, normally used for most of the standard communication protocols. The data stream supports multiple data width (byte/halfword/word) formats.

64.3 Main features

- Three contexts for the concurrent CRC computation are supported
- Separate CRC engine for each context
- Zero-wait states during the CRC computation (pipeline scheme)
- Three hard-wired polynomials (CRC-8, CRC-32 ITU-T V.42, and CRC-16-CCITT)
- Support for byte, halfword, or word width of the input data stream

64.3.1 Standard features

- Peripheral bus interface
- CRC-8 VDA CAN
- CRC-16-CCITT
- CRC-32 ITU-T V.42 (see note below)

NOTE

Although the CRC-32 ITU-T V.42 polynomial is used for CRC generation, the generation algorithm differs (MISR) from the one used by the IEEE 802.3 standard.

64.4 Block diagram

The top level diagram of the CRC module is given in the following figure. Refer to [Main features](#) for number of contexts in this module.

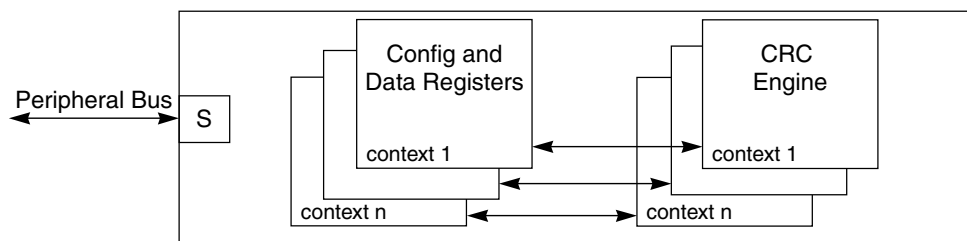


Figure 64-1. CRC top level diagram

64.5 External signal description

The CRC module does not generate any external signals.

64.5.1 Peripheral bus interface

The peripheral bus interface is a slave bus used for configuration and data streaming (CRC computation) purposes via CPU or DMA. The following bus operations (contiguous byte enables) are supported:

- Word (32 bits) data write/read operations to any registers

- Low and high halfword (16 bits, data[31:16] or data[15:0]) data write/read operations to any registers
- Byte (8 bits, data[31:24], data[23:16], data[15:8], or data[7:0]) data write/read operations to any registers
- Any other operation (free byte enables or other operations) must be avoided.

The CRC module generates a transfer error in the following cases:

- Any write/read access to the register addresses not mapped on the peripheral but included in the address space of the peripheral
- Any write/read operation different from byte/halfword/word (free byte enables or other operations) on each register

The registers of the CRC module are read/write accessible in each access mode:

- user
- supervisor

The following summarizes bus operation performance:

- Zero wait states (single bus cycle) for each write/read operation to the CRC_CFG and CRC_INP registers
- Zero wait states (single bus cycle) for each write operation to the CRC_CSTAT register
- Double wait states (3 bus cycles) for each read operation to the CRC_CSTAT or CRC_OUTP registers immediately following (next clock cycle) a write operation to the CRC_CSTAT, CRC_INP, or CRC_CFG registers belonging to the same context; in all the other cases, no wait states are inserted

64.6 CRC memory map and registers

CRC memory map

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/page
0	Configuration Register (CRC_CFG1)	32	R/W	See section	64.6.1/3254
4	Input Register (CRC_INP1)	32	R/W	0000_0000h	64.6.2/3255
8	Current Status Register (CRC_CSTAT1)	32	R/W	FFFF_FFFFh	64.6.3/3256
C	Output Register (CRC_OUTP1)	32	R	FFFF_FFFFh	64.6.4/3257

Table continues on the next page...

CRC memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
10	Configuration Register (CRC_CFG2)	32	R/W	See section	64.6.1/3254
14	Input Register (CRC_INP2)	32	R/W	0000_0000h	64.6.2/3255
18	Current Status Register (CRC_CSTAT2)	32	R/W	FFFF_FFFFh	64.6.3/3256
1C	Output Register (CRC_OUTP2)	32	R	FFFF_FFFFh	64.6.4/3257
20	Configuration Register (CRC_CFG3)	32	R/W	See section	64.6.1/3254
24	Input Register (CRC_INP3)	32	R/W	0000_0000h	64.6.2/3255
28	Current Status Register (CRC_CSTAT3)	32	R/W	FFFF_FFFFh	64.6.3/3256
2C	Output Register (CRC_OUTP3)	32	R	FFFF_FFFFh	64.6.4/3257

64.6.1 Configuration Register (CRC_CFGn)

Access: User read/write.

Address: 0h base + 0h offset + (16d × i), where i=0d to 2d

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0								0							
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31		
R	0										SWAP_BYTEWISE		SWAP_BITWISE		POLYG		SWAP	INV
W																		
Reset	0	0	0	0	0	0	0	0	0	0	0	0	*	*	0	0		

* Notes:

- POLYG field: Reset value is a function of n in CRC_CFGn. When $n\%2 = 0$, reset value = 01b. When $n\%2 = 1$, reset value = 00b. So CRC_CFG1 POLYG reset value = 00b (Since $1\%2=1$).

CRC_CFGn field descriptions

Field	Description
0–7 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

Table continues on the next page...

CRC_CFGn field descriptions (continued)

Field	Description
8–25 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
26 SWAP_ BYTEWISE	Swap CRC_INP byte-wise NOTE: INV and SWAP bits are set to 1 for CRC-32 polynomial calculations. 0 Do not swap 1 Perform byte-wise swap on CRC_INP input data internally for CRC-32 polynomial calculations.
27 SWAP_BITWISE	Swap CRC_INP bit-wise 0 Do not swap 1 Perform bit-wise swap on CRC_INP input data internally for CRC-8 and CRC-16 polynomial calculations.
28–29 POLYG	Polynomial selection This bit can be written only during the configuration phase. 00 CRC-CCITT polynomial 01 CRC-32 polynomial 10 CRC-8 polynomial 11 Reserved
30 SWAP	Swap selection This bit can be written only during the configuration phase. The swap operation is a bit-by-bit swapping of the content. 0 No swap selection applied on the CRC_OUTP content 1 Swap selection (MSB to LSB, LSB to MSB) applied on the CRC_OUTP content; in case of CRC-CCITT polynomial, the swap operation is applied on the 16 least-significant bits
31 INV	Inversion selection This bit can be written only during the configuration phase. The inversion operation is a complement (or negation) of the content. 0 No inversion selection applied on the CRC_OUTP content 1 Inversion selection (bit x bit) applied on the CRC_OUTP content

64.6.2 Input Register (CRC_INPn)

Access: User read/write.

NOTE

High level code must be written in a way to realize byte-, halfword- and word-wide writes on assembler code level. Any endianness problem faced can be corrected by appropriate configuration of CRC_CFG[SWAP_BYTEWISE] and CRC_CFG[SWAP_BITWISE].

CRC memory map and registers

Address: 0h base + 4h offset + (16d × i), where i=0d to 2d

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	INP																															
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	

CRC_INP*n* field descriptions

Field	Description
0–31 INP	Input data for the CRC computation This register can be written with word-, halfword- (high or low), or byte-wide writes in any sequence. Only the bits written are fed to the CRC engine. In case of halfword write operation, the bytes must be contiguous.

64.6.3 Current Status Register (CRC_CSTAT*n*)

Access: User read/write.

Address: 0h base + 8h offset + (16d × i), where i=0d to 2d

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	CSTAT																															
W																																
Reset	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	

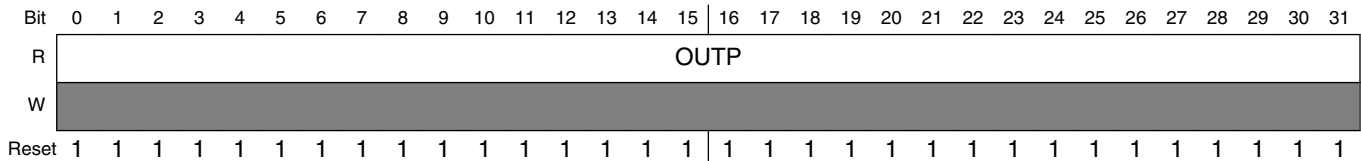
CRC_CSTAT*n* field descriptions

Field	Description
0–31 CSTAT	CRC signature status This register includes the current status of the CRC signature. No bit swap and inversion are applied to this register. In the case of the CRC-CCITT polynomial, only the 16 least-significant bits are significant. The 16 most-significant bits are set to 0 during the computation. In the case of the CRC-8 polynomial, only the 8 least-significant bits are significant. The 24 most-significant bits are set to 0 during the computation. The CSTAT register can be written at byte, halfword, or word. This register can be written only during the configuration phase.

64.6.4 Output Register (CRC_OUTP_n)

Access: User read-only.

Address: 0h base + Ch offset + (16d × i), where i=0d to 2d



CRC_OUTP_n field descriptions

Field	Description
0–31 OUTP	Final CRC signature This register includes the final signature corresponding to the CRC_CSTAT register value eventually swapped and inverted. In the case of the CRC-CCITT polynomial, only the 16 least-significant bits are significant. The 16 MSB bits are set to 0 during the computation. In the case of the CRC-8 polynomial, only the 8 least-significant bits are significant. The 24 most-significant bits are set to 0 during the computation.

64.7 Functional description

The CRC module supports the CRC computation for each context. Each context has its own complete set of registers, including the CRC engine. The data flow of each context can be interleaved. The data stream can be structured as a sequence of bytes, halfwords, or words. The input data sequence is provided, eventually mixing the data formats (byte, halfword, and word), writing to the input data register (CRC_INP).

The data stream is generally executed by *n* concurrent DMA data transfers (mem2mem) where *n* is less or equal to the number of contexts.

The standard generator polynomials are given in [Figure 64-18](#), [Figure 64-19](#), and [Figure 64-20](#) for the CRC computation of each context. Two separate registers are available: CRC_INP for writing data and CRC_CSTAT for retrieving the (accumulated up to now) CRC.

$$x^8 + x^4 + x^3 + x^2 + 1$$

Figure 64-18. CRC8 VDA CAN

$$x^{16} + x^{12} + x^5 + 1$$

Figure 64-19. CRC-CCITT (x.25 protocol)

$$x^{32}+x^{26}+x^{23}+x^{22}+x^{16}+x^{12}+x^{11}+x^{10}+x^8+x^7+x^5+x^4+x^2+x+1$$

Figure 64-20. CRC-32 (ITU-T V.42 protocol)

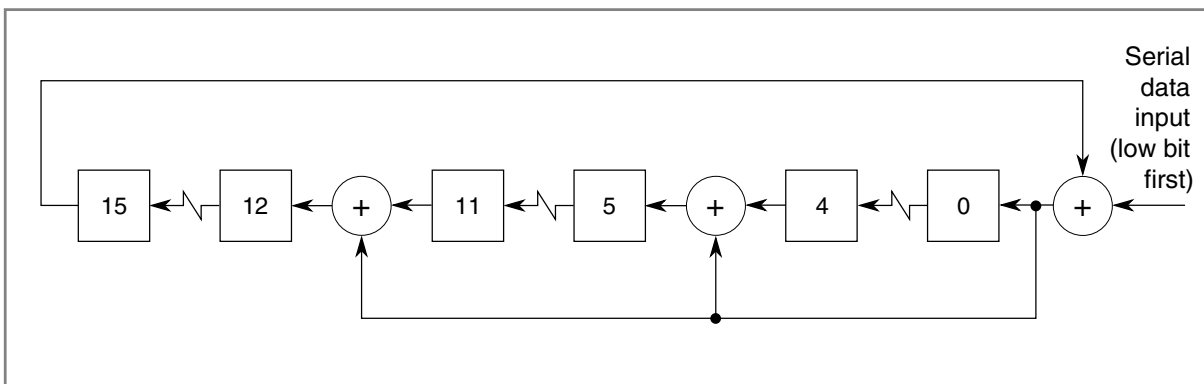


Figure 64-21. CRC-CCITT engine concept scheme

The initial seed value of the CRC can be programmed by initializing the CRC_CSTAT register. A diagram illustrating serial data loading of the CRC engine is shown in [Figure 64-21](#) for the CRC-CCITT. Note that, conceptually, the least-significant, or "right-most," bit shown in [Input Register \(CRC_INP_n\)](#), is fed first into the engine. The actual implementation executes the CRC computation in a single clock cycle (parallel data loading). A pipeline scheme has been adopted to decouple the IPS bus interface from the CRC engine in order to allow the computation of the CRC at speed (zero wait states).

If the CRC signature is used for encapsulation in the data frame of a communication protocol (for example, SPI, etc.), a bit swap (high bit for low bit or low bit for high bit) and/or bit inversion of the final CRC signature can be applied to CRC_OUTP before transmitting the CRC.

Use of the CRC module is summarized in the flow chart given in [Figure 64-22](#).

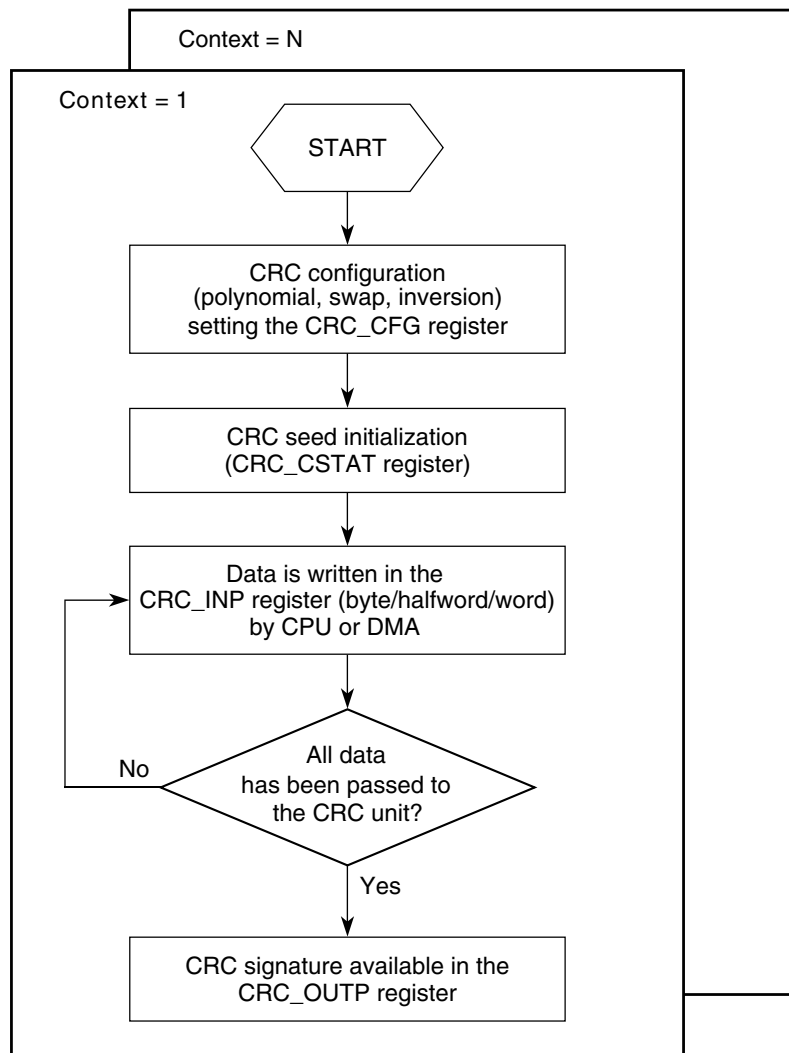


Figure 64-22. CRC computation flow

64.8 Use cases

64.8.1 Programming example

The number of contexts depends on the application. The overall number of contexts for the CRC peripheral depends on the number of peripherals that concurrently require intervention by the CRC module. Two main use cases are considered:

- Calculation of the CRC of the configuration registers during the process safety time
- Calculation of the CRC on the incoming/outcoming frames for the communication protocols (not protected with CRC by definition of the protocol itself) used as a safety-relevant peripheral

The signature of the configuration registers is computed correctly only if these registers do not contain any status bit.

Assuming that the DMA engine has n channels (greater than or equal to the number of contexts) configurable for the following types of data transfer—mem2mem, periph2mem, mem2periph—the following sequence, as shown in the following figure, is applied to manage the transmission data flow:

1. DMA/CRC module configuration (context x , channel x) by CPU
2. Payload transfer from the MEM to the CRC module (CRC_INP register) after MSB-to-LSB change (input mirroring) to calculate the CRC signature (phase1) by DMA (mem2mem data transfer, channel x)
3. CRC signature copy from the CRC module (CRC_OUTP register) to the MEM (phase 2) by CPU
4. Data block (payload + CRC) transfer from the MEM to the PERIPH module (for example, SPI Tx FIFO) (phase 3) by DMA (mem2periph data transfer, channel x)

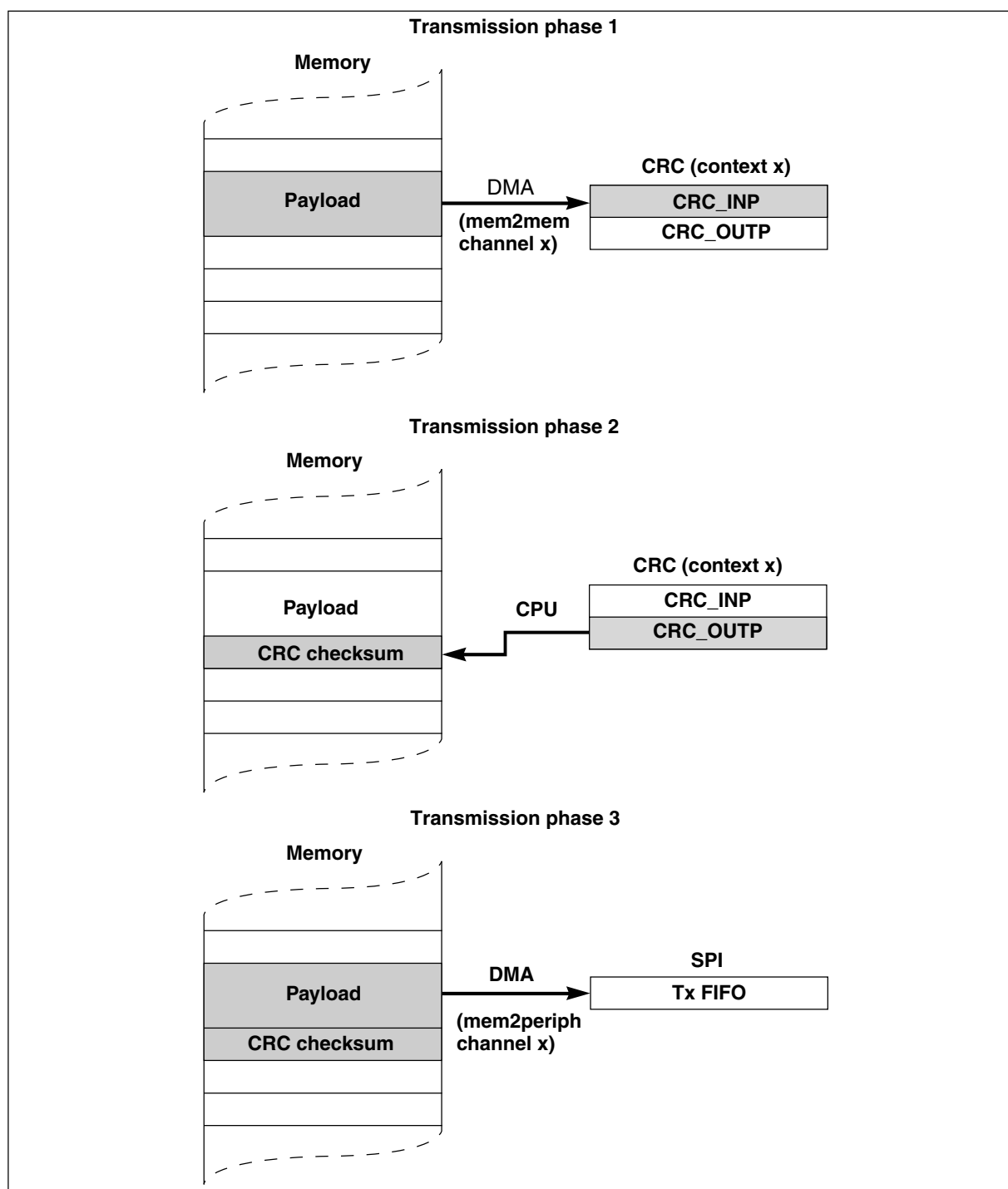


Figure 64-23. DMA-CRC transmission sequence

64.8.2 Register programming

- INV and/or SWAP (affecting the output only) can be applied to CRC-8, CRC-16, and CRC-32. Both can be applied together.

Use cases

- SWAP_BITWISE (affecting the input only) can be applied for all polynomials if required by an application.
- SWAP_BYTEWISE (affecting the input only) can be applied for CRC-16 and CRC-32 polynomials only.
- SWAP_BITWISE has priority over SWAP_BYTEWISE when both are applied together.
- When generating CRC-32 for the ITU-T V.42 standard the user needs to set SWAP_BYTEWISE together with INV and SWAP.
- When generating CRC-16 the user needs to set SWAP_BITWISE bit.

Chapter 65

Security Overview

65.1 Introduction

This chip has a comprehensive set of customer-configurable security features designed to protect code and data from unauthorized access.

65.2 Security features summary

The chip's security features include:

- Device censorship based on life cycle model
- Memory security:
 - Password protection for flash memory with PASS module
 - One-time-programmable (OTP) flash memory areas
 - Tamper detection including flash memory erase counter
 - MBIST functionality that can be enabled for execution during an STCU2 off-line condition to initialize RAMs (see the STCU2 chapter for details about MBIST execution in an STCU2 off-line condition)
 - BAM with software routines to initialize RAMs when not booting from internal flash memory
 - Security functions described in the *Secure Hardware Extension (SHE) Functional Specification Version 1.1*
- Monitoring of operating conditions
- Security watchdog
- Secure debug

65.3 Flash memory read, write, and erase protection

[Flash memory read protection scenarios](#) explains the supported options for read protection.

The C55FMC and PASS modules provide write and erase protection:

- The C55FMC_LOCK n registers provide one level of write and erase protection.
- The PASS_LOCK n _PG m registers provide a higher level of password-protected write and erase protection that supersedes C55FMC's protection by overwriting the values of the C55FMC_LOCK n registers.
- After bootup, during which the values of PASS_LOCK n _PG m registers are loaded from DCF records: Users can override these values temporarily, until the next reset, by programming them to different values, thereby also temporarily programming different values for the C55FMC_LOCK n registers.

For details about these registers, see the [Embedded Flash Memory \(C55FMC\)](#) and [Password and Device Security Module \(PASS\)](#) chapters.

Chapter 66

Cryptographic Services Engine (CSE)

66.1 Chip-specific CSE information

This chip has one instance of the CSE module. The module:

- executes the chip's secure boot process. See [Parallel secure boot](#) in the System Boot details.
- has exclusive access to the flash memory blocks mapped to the C55FMC_LOCK1 register, PASS_LOCK1_PGn registers, and TDRn_LOCK1 DCF client. See [C55FMC_LOCK1 register bit mapping](#). The system MPU is automatically configured to prevent other bus masters from interfering with CSE's access to the flash memory.

66.1.1 CSE configuration and SECURE_BOOT execution

The [UTEST Secure Boot Configurations DCF client](#) automatically configures aspects of CSE operation at startup.

See this DCF client's description for details about its factory-programmed value, which ensures that reset logic issues the SECURE_BOOT command to CSE during reset.

66.1.2 Sys Clk Freq for CSE

In the description of CSE_CR[DIV], the mentioned 'Sys Clk Freq' corresponds to the plat_clk frequency on this chip. See the Clocking chapter for more information about plat_clk.

66.1.3 CSE clock: minimum frequency

To ensure compliance with SHE specification performance requirements, the CSE clock source (FM_PER_CLK) must have a minimum frequency of 105 MHz.

66.1.4 CSE debug not supported

This chip has no debug capability for the CSE module. Descriptions of CSE debug features in this chapter do not apply.

66.1.5 CSE command-processing completion before Stop mode entry

To ensure proper CSE operation: Before setting SIU_HLT1[CSE] to 1, ensure CSE_SR[BSY] is 0.

66.2 Introduction

66.2.1 Overview

The Cryptographic Services Engine (CSE) is a peripheral module that implements the security functions described in the *Secure Hardware Extension (SHE) Functional Specification Version 1.1*.

The CSE design includes a host interface with a set of memory mapped registers that are used by the CPU to issue commands and a system bus interface that allows the CSE to directly access system memory. Two dedicated blocks of system flash memory are used by the CSE for secure key storage.

66.2.2 Features

The CSE has the following features:

- Secure storage for cryptographic keys
- AES-128 encryption and decryption
- AES-128 CMAC authentication

- True random number generation
- Secure boot mode
- System bus master interface

66.2.3 Modes of operation

The CSE supports operation in Normal and Debug modes of operation. The use of the cryptographic keys stored by the CSE is controlled based on the activation of the CPU debug port and the successful completion of the secure boot process.

The CSE has a low-power mode that disables the clock to all logic except the host interface. Register accesses are supported in this mode, but commands are not processed.

66.2.4 Block diagram

The CSE design includes a command processor, host interface, system bus interface, local memory, AES logic, and True Random Number Generator (TRNG) as shown below.

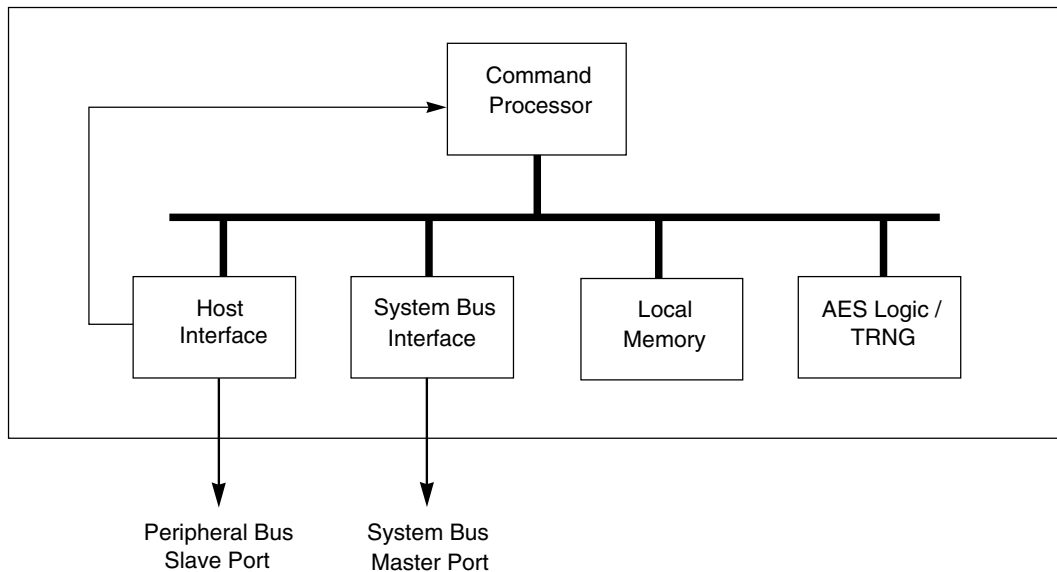


Figure 66-1. CSE block diagram

66.3 External signal description

The CSE has no external interface signals.

66.4 Memory Map and Register Definition

The CSE programming model has ten 32-bit registers. The programming model can only be accessed using 32-bit (word) accesses. References using a different size are invalid. The CSE memory map is shown in the table below.

CSE memory map

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
0	CSE Control Register (CSE_CR)	32	R/W	See section	66.4.1/3268
4	CSE Status Register (CSE_SR)	32	R	0000_0000h	66.4.2/3270
8	CSE Interrupt Register (CSE_IR)	32	w1c	See section	66.4.3/3271
C	CSE Error Code Register (CSE_ECR)	32	R	0000_0000h	66.4.4/3272
20	CSE Command Register (CSE_CMD)	32	R/W	0000_0000h	66.4.5/3273
24	CSE Parameter Register (CSE_P1)	32	R/W	0000_0000h	66.4.6/3274
28	CSE Parameter Register (CSE_P2)	32	R/W	0000_0000h	66.4.6/3274
2C	CSE Parameter Register (CSE_P3)	32	R/W	0000_0000h	66.4.6/3274
30	CSE Parameter Register (CSE_P4)	32	R/W	0000_0000h	66.4.6/3274
34	CSE Parameter Register (CSE_P5)	32	R/W	0000_0000h	66.4.6/3274

66.4.1 CSE Control Register (CSE_CR)

The CSE_CR contains fields for configuring and controlling operation of the CSE.

Address: 0h base + 0h offset = 0h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	DIV								0							
W											KBS	SFE	MDIS	SUS	DRE	CIE
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	*

* Notes:

- CIE field: The reset value of this bit is 0 on POR, but when the reset lifts and CSE execution starts, the value of this bit becomes 1. So, after the first read access to this bit, the read value is 1.

CSE_CR field descriptions

Field	Description
0–15 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
16–23 DIV	TRNG Clock Divider Select. Sets the clock divide ratio for the TRNG clock. The TRNG clock is the system clock divided by a programmable ratio: $\text{TRNG Clk Freq} = \text{Sys Clk Freq} / (2 * (\text{DIV} + 1))$ The divide ratio must be set such that the TRNG clock frequency is between 500 kHz and 2MHz. 0x00 divide by 2 0x01 divide by 4 0x02 divide by 6 ... 0xFE divide by 510 0xFF divide by 512
24–25 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
26 KBS	Key Bank Select. Selects which set of application keys are visible for commands. See Table 66-13 . 0 Select KEY_1-KEY_10 1 Select KEY_11-KEY_20
27 SFE	Security Flag Extension. When the SFE bit is set, the CSE recognizes the VERIFY_ONLY security flag during LOAD_KEY processing; otherwise, it is ignored. The SFE bit has no effect on the enforcement of the VERIFY_ONLY security flag once a key is loaded. 0 Security Flag Extension disabled. 1 Security Flag Extension enabled.
28 MDIS	Module Disable. When the MDIS bit is set, the CSE is put into a low-power mode that disables the clock to all of the CSE logic except for the host interface. The MDIS bit should not be set during command processing (CSE_SR[BSY] = 1). The current command should be canceled and processing stopped (CSE_SR[BSY] = 0) before setting the MDIS bit. 0 Normal mode 1 Low-power mode
29 SUS	Suspend Command Processing. When the SUS bit is set, the CSE suspends processing of the current command until the SUS bit is cleared. The current execution status of the command processor is reflected by the CSE_SR[EX] flag. 0 Enable processing of commands. 1 Suspend command processing.
30 DRE	DMA Request Enable. When the DRE bit is set, the DMA request is asserted when command processing is completed. NOTE: DMA is not implemented on this chip. 0 DMA request not enabled. 1 DMA request enabled.
31 CIE	Command Complete Interrupt Enable. When the CIE bit is set, an interrupt request is generated and the CSE_IR[CIF] flag is set when command processing is completed. 0 Command Complete Interrupt disabled. 1 Command Complete Interrupt enabled.

66.4.2 CSE Status Register (CSE_SR)

The CSE_SR contains flags indicating the status of the CSE. Reading CSE_SR[24:31] is the same as the SHE GET_STATUS command.

Address: 0h base + 4h offset = 4h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	
R	0																
W																	
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	
R	0								EX	IDB	EDB	RIN	BOK	BFN	BIN	SB	BSY
W																	
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	

CSE_SR field descriptions

Field	Description
0–22 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
23 EX	Execute. NOTE: Ignore the status of CSE_SR[EX] when CSE_SR[BSY] flag is '0'. 0 Command processor is idle or suspended. 1 Command processor is executing a command.
24 IDB	Internal Debug. The IDB bit is set by the SECURE_BOOT, INIT_CSE, and DEBUG_AUTH commands when no user keys (i.e. all keys except SECRET_KEY and UID) are stored. It is cleared on reset and by the LOAD_KEY command. 0 Internal Debug functions disabled. 1 Internal Debug functions enabled.
25 EDB	External Debug. The EDB bit is set when the CPU debug port is activated and is cleared on reset. 0 External debugger not attached. 1 External debugger attached.
26 RIN	Random Number Generator Initialized. The RIN bit is set by the INIT_RNG command and is cleared on reset and by the DEBUG_AUTH command. 0 Random number generator not initialized. 1 Random number generator initialized.
27 BOK	Secure Boot OK. The BOK bit is set by the successful completion of the SECURE_BOOT command. It is cleared by reset and by the BOOT_FAILURE command.

Table continues on the next page...

CSE_SR field descriptions (continued)

Field	Description
	0 Secure boot not completed or secure boot failure. 1 Secure boot successful.
28 BFN	Secure Boot Finished. The BFN bit is set by the SECURE_BOOT command when the BIN bit is set, when an error is encountered, or when the BOOT_MAC value does not match. It is also set by the BOOT_OK or BOOT_FAILURE commands. It is cleared on reset. The BFN bit is not set in the event the CSE flash blocks are accessible, but contain an invalid firmware image as detected by a checksum mismatch (CSR_ECR=0x13). 0 Secure boot not finished. 1 Secure boot finished.
29 BIN	Secure Boot Initialization. The BIN bit is set by the SECURE_BOOT command if the BOOT_MAC memory slot is empty and is cleared on reset. 0 Secure boot personalization not completed. 1 Secure boot personalization completed.
30 SB	Secure Boot. The SB bit is set by the SECURE_BOOT command if the BOOT_MAC_KEY slot is not empty, and is cleared on reset. 0 Secure boot not activated. 1 Secure boot activated.
31 BSY	Busy. The BSY bit is set when a command is issued and cleared when command processing is completed. 0 Command processing completed. 1 Command processing not completed.

66.4.3 CSE Interrupt Register (CSE_IR)

The CSE_IR contains the command completion interrupt flag bit.

Address: 0h base + 8h offset = 8h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0															CIF
W																w1c
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	*

* Notes:

- CIF field: The reset value of this bit is 0 on POR, but when the device boots in a mode that supports SECURE_BOOT, the value of this bit becomes 1 upon the command's completion.

CSE_IR field descriptions

Field	Description
0–30 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
31 CIF	Command Complete Interrupt Flag. The CIF flag reflects the state of the command complete interrupt request. The CIF flag and interrupt request are cleared by writing a 1 to this bit. Writing a 0 has no effect. 0 No interrupt request. 1 Interrupt request due to completion of a command.

66.4.4 CSE Error Code Register (CSE_ECR)

The CSE_ECR contains the error code from the last completed command.

Address: 0h base + Ch offset = Ch

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0																EC															
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	

CSE_ECR field descriptions

Field	Description
0–26 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
27–31 EC	Error Code from the last command completed. 0x00 No error 0x02 Command sequence error 0x03 Key not available 0x04 Invalid key 0x05 Empty key 0x06 No secure boot 0x07 Key write protected 0x08 Key update error 0x09 Random number seed not initialized 0x0A Internal debug not allowed 0x0B Command issued while busy 0x0C System memory error 0x10 Internal memory error 0x11 Invalid command

Table continues on the next page...

CSE_ECR field descriptions (continued)

Field	Description
	0x12 TRNG error
	0x13 CSE flash block error
	0x14 Internal command processor error
	0x15 Length error

66.4.5 CSE Command Register (CSE_CMD)

Commands are issued by first loading the appropriate parameter registers (CSE_Px) and then writing a command code to the command register (CSE_CMD). Reads of the CSE_CMD register have no effect on the operation of the CSE. See [Command processing](#) for a description of command processing and [CSE commands](#) for a description of each command.

Address: 0h base + 20h offset = 20h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0																										CMD					
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

CSE_CMD field descriptions

Field	Description
0–26 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
27–31 CMD	Command. See CSE commands for details of each command. 0x01 = ENC_ECB 0x02 = ENC_CBC 0x03 = DEC_ECB 0x04 = DEC_CBC 0x05 = GENERATE_MAC 0x06 = VERIFY_MAC 0x07 = LOAD_KEY 0x08 = LOAD_PLAIN_KEY 0x09 = EXPORT_RAM_KEY 0x0A = INIT_RNG 0x0B = EXTEND_SEED 0x0C = RND

Table continues on the next page...

CSE_CMD field descriptions (continued)

Field	Description
	0x0D = SECURE_BOOT
	0x0E = BOOT_FAILURE
	0x0F = BOOT_OK
	0x10 = GET_ID
	0x11 = CANCEL
	0x12 = DEBUG_CHAL
	0x13 = DEBUG_AUTH
	0x14 = TRNG_RND
	0x15 = INIT_CSE
	0x16 = MP_COMPRESS

66.4.6 CSE Parameter Register (CSE_Pn)

The five parameter registers (CSE_Px) are used to hold command parameter values. See [Command processing](#) for a description of how to issue commands and [CSE commands](#) for a description of each command. The parameter registers are read only when the CSE_SR[BSY] bit is set.

Address: 0h base + 24h offset + (4d × i), where i=0d to 4d

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	PARM																															
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	

CSE_Pn field descriptions

Field	Description
0–31 PARM	Command parameter (data value or address of data value).

66.5 CSE functional description

The CSE implements a comprehensive set of cryptographic functions as described in the *SHE Functional Specification*, including secure key storage, AES encryption, secure boot, AES CMAC authentication, and random number generation.

66.5.1 Host interface

The host interface includes all of the CSE memory mapped registers (see the memory map and register section).

The Control Register (CSE_CR) is used to set configuration options and control operation of the CSE. If the CSE_CR[CIE] bit is set, an interrupt request is generated after the CSE finishes processing each command. The interrupt is cleared by writing a 1 to the CSE_IR[CIF] bit. When the CSE_CR[SUS] bit is set, the CSE suspends processing of the current command and does not respond to new commands including the CANCEL command.

Command processing is resumed by clearing the CSE_CR[SUS] bit. The CSE cannot respond immediately to changes in the CSE_CR[SUS] bit, so the CSE_SR[EX] flag indicates the current status of the CSE command processor. The CSE_SR[EX] flag is set when the CSE command processor is running and is cleared when the CSE is idle or has suspended processing. The CSE_CR[MDIS] bit puts the CSE into a low-power mode. In low-power mode, registers can be accessed but commands are not processed. Before setting the CSE_CR[MDIS] bit or entering any device low-power modes, application software should cancel any pending command and wait for the CSE_SR[BSY] bit to be cleared. The CSE_CR[SFE] bit enables loading of the VERIFY_ONLY flag during the LOAD_KEY command. The CSE_CR[KBS] bit selects which key bank (KEY_1–KEY_10 or KEY_11–KEY_20) is active.

The Status Register (CSE_SR) contains a set of nine status flags. Reading CSE_SR[24:31] is equivalent to the SHE GET_STATUS command. The CSE_SR[BSY] bit indicates when the CSE is processing a command. The flag is cleared when processing is completed. Polling the CSE_SR[BSY] bit is an alternative to interrupt-driven operation. The CSE_SR[SB], CSE_SR[BIN], CSE_SR[BFN] and CSE_SR[BOK] bits reflect the secure boot status (see [Secure boot](#)). The CSE_SR[RIN] bit is set when the PRNG is initialized with a seed value generated by the TRNG. The CSE_SR[EDB] is set if an external debugger is connected. The CSE_SR[IDB] flag is set when internal debugging has been enabled with the DEBUG_CHAL and DEBUG_AUTH commands. The CSE_SR is a read-only register and can be read at any time.

The Error Code Register (CSE_ECR) contains the error code from the last completed command. The CSE_ECR is set to 0 if no error occurred while processing the command. The CSE_ECR is a read-only register.

66.5.2 Command processing

CSE functions are accessed through a set of commands that are listed in [CSE commands](#). Commands are issued by first loading the appropriate parameter registers (CSE_Px) and then writing a command code to the command register (CSE_CMD). CSE_CMD register reads have no effect on the operation of the CSE.

Command inputs and outputs that are 32 bits or less in size are stored directly into a parameter register. Inputs and outputs that are larger than 32 bits are kept in system memory and the address of the data (pointer) is loaded into the parameter register. All data must be aligned on a 32-bit (word) boundary. The two least significant bits of an address are ignored. All data is stored in big-endian format with the most significant byte of the data block stored in the lowest address.

The first command issued to the CSE after reset must be either SECURE_BOOT or INIT_CSE. In some device boot modes, system boot logic or BAM code will issue the SECURE_BOOT command, which initializes the CSE and starts the SHE secure boot protocol (see [Secure boot](#)). For details about how and when the SECURE_BOOT command is issued on this chip, see the chip-specific CSE information. For the other boot modes, application software must issue the INIT_CSE command before issuing other CSE commands. The INIT_CSE command can only be issued one time after reset and only in boot modes that do not issue the SECURE_BOOT command. Application software cannot issue the SECURE_BOOT command.

The CSE can only process one command at a time. Commands take a number of clock cycles to complete, so one of the following must be used to determine when the output data is valid and the next command can be issued:

- The CSE_SR[BSY] flag
- The interrupt request

Parameter registers cannot be modified while the CSE is processing a command. Command processing can be suspended and resumed using the CSE_CR[SUS] bit (see [Host interface](#)). If the CSE_CMD register is written while a command is being processed, processing is aborted and an error is generated with CSE_ECR[EC] = 0x0B. The CANCEL command can be used to abort processing of a command without generating an error. The CSE_ECR is updated and the CSE_SR[BSY] bit is cleared upon completion of command processing (see [Error handling](#)).

66.5.3 Secure storage

The CSE provides secure, nonvolatile storage for cryptographic keys as described in the *SHE Functional Specification*. The keys are stored in 25 memory slots, with one ROM slot, 23 nonvolatile slots, and one RAM slot as shown in [Table 66-13](#). The first four slots have a dedicated use; the other slots are available for application specific keys. The CSE extends the SHE specification by providing 10 extra application key slots (KEY_11–KEY_20) which are enabled by setting the CSE_CR[KBS] bit. The BOOT_MAC slot is loaded with a MAC value used by the secure boot process. All other slots are used for encryption or message authentication keys. The SECRET_KEY slot is programmed with a random value during device fabrication. All CSE encryption and message authentication commands specify a key by its Key ID.

Table 66-13. Memory slots

Slot name	KBS	Key ID	Type
SECRET_KEY	x	0x0	ROM
MASTER_ECU_KEY	x	0x1	nonvolatile
BOOT_MAC_KEY	x	0x2	nonvolatile
BOOT_MAC	x	0x3	nonvolatile
KEY_1–KEY_10	0	0x4–0xD	nonvolatile
KEY_11–KEY_20	1	0x4–0xD	nonvolatile
RAM_KEY	x	0xE	RAM

In addition to the 25 memory slots, the CSE holds a 120-bit read-only unique identification number (UID) that is programmed during device fabrication. The UID is used in the memory update procedure and is available for application-specific uses.

Each memory slot holds a 128-bit value, 6 security flags, and a 28-bit counter. See the *SHE Functional Specification* for the required structure needed for the LOAD_KEY command. The security flags are defined in [Table 66-14](#). The 28-bit counter must be advanced each time a new key is loaded. The counter value is zero for an empty memory slot.

Table 66-14. Memory slot security flags

Flag name	Description
VERIFY_ONLY	If set, the memory slot key cannot be used by the GENERATE_MAC command, only by the VERIFY_MAC command. This bit has no effect if KEY_USAGE = 0.
WRITE_PROT	If set, the memory slot cannot be updated.
BOOT_PROT	If set, the memory slot is disabled if CSE_SR[BOK] = 0 or CSE_SR[BFN] = 0.
DEBUG_PROT	If set, the memory slot is disabled if CSE_SR[EDB] = 1.
KEY_USAGE	If set, the memory slot holds a MAC key; otherwise, it holds an encryption key.
WILDCARD	If set, the memory slot cannot be updated with the wildcard UID.

The memory slot security flags for a specified key are checked against the associated bits in the CSE_SR during command processing as described in the SHE specification. If the memory slot is disabled due to security flag settings, a key not available error (EC = 0x03) is returned by the command. The VERIFY_ONLY security flag is a CSE-specific feature that is not defined in the SHE specification. If the CSE_CR[SFE] bit is set when the LOAD_KEY command is executed, the security flag field (FID) in the M2 message is extended by one bit with the VERIFY_ONLY flag and the adjacent zero fill field is reduced to 94 bits. The CSE_CR[SFE] bit has no effect on the enforcement of the VERIFY_ONLY flag for commands issued after the LOAD_KEY.

A key is loaded into a memory slot using the LOAD_KEY command which implements the SHE memory update protocol. An empty key slot can only be used as the authorization key to update itself (KeyID = AuthID). In this case, the authorization key has a value of 0. The BOOT_PROT and DEBUG_PROT security flags are not enforced for the LOAD_KEY command except when the RAM_KEY slot is loaded. In that case, the BOOT_PROT and DEBUG_PROT flags of the AuthID slot are applied. Additionally, when loading the RAM_KEY slot, the counter and security flag fields in the M2 message must be zero.

The LOAD_PLAIN_KEY command can be used to load a key into the RAM_KEY slot. The EXPORT_RAM_KEY command can then be used to export the key in an encrypted form compatible with the LOAD_KEY command for storage outside the CSE. The BOOT_PROT and DEBUG_PROT security flags of the MASTER_ECU_KEY are enforced for the EXPORT_RAM_KEY command.

Two dedicated blocks in the system flash memory are used to implement the secure storage feature. These blocks are not program visible and only the CSE can read, erase, and program these blocks. The CSE reads and writes to the system flash memory via the system bus master interface during the SECURE_BOOT, INIT_CSE, DEBUG_AUTH, and LOAD_KEY commands. During execution of these commands, the system MPU (if present) must be configured to allow the CSE access to the system flash memory and other bus masters must be programmed properly to avoid interfering with the CSE.

66.5.4 Encryption and decryption

The CSE supports AES-128 encryption and decryption in ECB and CBC modes of operation. The key is selected from one of the memory slots, which must be enabled for the operation. A plaintext key can be loaded into the RAM_KEY slot using the LOAD_PLAIN_KEY command for keys that are not stored in a nonvolatile memory slot.

66.5.5 Message authentication

The CSE uses the AES-128 CMAC algorithm for message authentication. The key for the CMAC operation is selected from one of the memory slots which must be enabled for the operation. A plaintext key can be loaded into the RAM_KEY slot using the LOAD_PLAIN_KEY command for keys that are not stored in a nonvolatile memory slot. The VERIFY_MAC command supports comparison of a calculated MAC with an input MAC value.

66.5.6 Secure boot

The CSE implements the SHE secure boot protocol. In some device boot modes, the BAM or system boot logic issues the SECURE_BOOT command to the CSE, which starts the secure boot process. For details about how and when the SECURE_BOOT command is issued on this chip, see the chip-specific CSE information.

The first step in the process is for the CSE to download the command processor firmware and memory slot data from the CSE flash memory blocks into local memory. If the BOOT_MAC_KEY slot is empty, the CSE_SR[SB] flag is cleared and the process is finished. Otherwise, the CSE_SR[SB] flag is set and CSE calculates the MAC over the specified bootloader code. If the BOOT_MAC slot is empty, the calculated MAC is loaded into the BOOT_MAC slot, the CSE_SR[BIN] and CSE_SR[BFN] flags are set, and the process is finished. Otherwise, the calculated MAC value is compared to the value in the BOOT_MAC slot. If the values match, the CSE_SR[BOK] bit is set; otherwise, the CSE_SR[BFN] bit is set. If the CSE_SR[BOK] flag is set, the user boot code can issue the BOOK_OK command which sets the CSE_SR[BFN] bit. The memory slots that have the BOOT_PROT flag set are enabled when both the CSE_SR[BFN] and CSE_SR[BOK] flags are set.

The SECURE_BOOT command can run either before the user boot code is executed or in parallel with the user boot code. If the parallel mode is used, the user boot code must suspend command processing (set CSE_CR[SUS] to 1) while performing any operations—such as configuring the flash controller—that may interfere with the CSE accessing flash memory. A device-specific bit selects whether to run the SECURE_BOOT command before or in parallel with the user boot code. For details about this bit, see the chip-specific CSE information.

The CSE implements an internal timer that is started when the SECURE_BOOT command is issued. The initial value of the timer is set to 0x00F4_2400 (16,000,000 CSE system clocks). If the timer expires before the SECURE_BOOT command completes, any application code intended to be blocked by the SECURE_BOOT process will instead see the non-completed SECURE_BOOT status of CSE_SR[BSY]=1 and

CSE_SR[BOK]=0. In the event of a SECURE_BOOT timeout, the CSE will continue executing the SECURE_BOOT command until completion, until suspended, or until receiving the CANCEL command from the application code. There is no error code associated with a SECURE_BOOT timeout.

66.5.7 Random number generation

The CSE has both a Pseudo Random Number Generator (PRNG) and a True Random Number Generator (TRNG).

The PRNG has a 128-bit state variable and uses AES in output feedback mode to generate pseudo random values. A key derived from the SECRET_KEY is used for the PRNG. The RND command updates the state of the PRNG and returns the 128-bit random value. The EXTEND_SEED command can be used to add entropy to the PRNG state. The PRNG state is initialized after each reset with the INIT_RNG command, which uses the TRNG to generate a 128-bit seed value for the PRNG. The CSE_SR[RIN] flag is set when the PRNG is initialized.

The INIT_RNG and TRNG_RND commands use the TRNG to generate truly random values. The TRNG hardware runs off of a slower clock derived from the system clock. The CSE_CR[DIV] field needs to be configured for these commands such that the TRNG clock is between 500 kHz and 2 MHz. Random values generated by the TRNG are checked with a statistical test to verify proper operation of the TRNG. If the test fails, a TRNG error (EC = 0x12) is returned. Due to the statistical nature of this test, there is a very small probability ($< 10^{-9}$) that a properly operating TRNG will return an error. If an TRNG error is returned, the command can be issued again.

66.5.8 Error handling

When the CSE command processor encounters an error condition, it stops processing and returns an error code as described in [Table 66-15](#). The CSE_SR[BSY] and CSE_SR[EX] bits are cleared with interrupt requests generated the same as if the command had completed successfully. In most cases, error conditions are detected before data processing begins and no output values are written. Intermediate or invalid output data is never written to the parameter registers or system memory. However, it is possible for outputs to system memory to be partially written when an error occurs. The CSE does not zero out this partially written data.

Table 66-15. Error code summary

Error code	Error description	Error conditions
0x00	No error	1. Command successfully executed with no errors encountered.
0x02	Command sequence error	1. Command issued (except for CANCEL) before the SECURE_BOOT or INIT_CSE command. 2. The DEBUG_AUTH command issued before the DEBUG_CHAL command. 3. SECURE_BOOT or INIT_CSE command issued after an initial SECURE_BOOT or INIT_CSE is issued.
0x03	Key not available	1. A required key is not available due to a BOOT_PROT or DEBUG_PROT security flag restriction (see Secure storage).
0x04	Invalid key	1. The specified key slot is not valid for the command. (See the SHE specification.) 2. The specified key slot is not available due to a KEY_USAGE security flag restriction (see Secure storage).
0x05	Empty key	1. The specified key slot is empty.
0x06	No secure boot	1. SECURE_BOOT issued when secure boot is disabled or the BOOT_MAC_KEY slot is empty. 2. BOOT_FAIL or BOOT_OK issued with incorrect settings for either the CSE_SR[SB], CSE_SR[BFN] or CSE_SR[BOK] flags.
0x07	Key write protected	1. Attempt to load a key slot with the WRITE_PROT security flag set. 2. Attempt to enter internal debug mode (DEBUG_CHAL, DEBUG_AUTH) with a WRITE_PROT security flag set in one or more key slots.
0x08	Key update error	The LOAD_KEY command failed due to one of the following conditions: 1. The M3 MAC value is invalid. 2. The wildcard UID is specified with the WILDCARD flag set. 3. The specified UID does not match the device UID. 4. The update counter or security flag values are not zero when loading the RAM_KEY slot. 5. The specified update counter value is not greater than the current counter value for the slot. (Counter value is zero for an empty slot.)
0x09	Random number seed not initialized	1. RND, EXTEND_SEED, or DEBUG_CHAL command issued before the INIT_RNG command.
0x0A	Internal debug not allowed	1. DEBUG_AUTH command issued with invalid MAC value.
0x0B	Command issued while busy	1. Command issued when the CSE_SR[BSY] bit is set.
0x0C	System memory error	1. A system memory error was encountered while executing the command. (The CSE flash memory block error code is generated for bus errors encountered when accessing the CSE flash memory blocks.)
0x10	Internal memory error	1. An internal memory error was encountered while executing the command.

Table continues on the next page...

Table 66-15. Error code summary (continued)

Error code	Error description	Error conditions
0x11	Invalid command	1. Value written to CSE_CMD register is out of range.
0x12	TRNG error	1. One or more statistical tests run on the TRNG output failed (see Generate Random Number).
0x13	CSE flash memory block error	1. Error reading, programming, or erasing one of the CSE flash memory blocks. 2. UID or SECRET_KEY required but not available.
0x14	Internal command processor error	1. An internal error condition was encountered while executing the command.
0x15	Length error	1. MAC length for VERIFY_MAC command is greater than 128. 2. Message length for GENERATE_MAC, VERIFY_MAC, or MP_COMPRESS command is greater than 0x7FFFFFFF (4 GB).

66.6 CSE commands

This section describes the set of CSE commands. Commands are issued by first loading the appropriate parameter registers (CSE_Px) and then writing a command code to the command register (CSE_CMD) as described in [Command processing](#). The first command issued to the CSE after reset must be INIT_CSE for device boot modes that do not support secure boot. Command inputs and outputs that are 32 bits or less in size are stored directly in a parameter register. Inputs and outputs that are larger than 32 bits are kept in system memory and the address of the data (pointer) is loaded into the parameter register. The data direction indication in the tables below refers to the command parameter value which may be stored in memory.

66.6.1 Encrypt ECB

The ENC_ECB command performs AES-128 encryption in ECB mode on n 128-bit blocks of data with the parameters specified in the table below.

Table 66-16. ENC_ECB command

Register	Value	Data direction
CSE_CMD	0x01	—
CSE_P1	Key ID	Input
CSE_P2	Number of blocks (n)	Input

Table continues on the next page...

Table 66-16. ENC_ECB command (continued)

Register	Value	Data direction
CSE_P3	First plaintext block address	Input
CSE_P4	First ciphertext block address	Output

66.6.2 Encrypt CBC

The ENC_CBC command performs AES-128 encryption on n 128-bit blocks of data in CBC mode with the parameters specified in the table below. The number of blocks parameter is a 32-bit value.

Table 66-17. ENC_CBC command

Register	Value	Data direction
CSE_CMD	0x02	—
CSE_P1	Key ID	Input
CSE_P2	IV address	Input
CSE_P3	Number of blocks (n)	Input
CSE_P4	First plaintext block address	Input
CSE_P5	First ciphertext block address	Output

66.6.3 Decrypt ECB

The DEC_ECB command performs AES-128 ECB decryption on n 128-bit blocks of data with the parameters specified in the table below.

Table 66-18. DEC_ECB command

Register	Value	Data direction
CSE_CMD	0x03	—
CSE_P1	Key ID	Input
CSE_P2	Number of blocks (n)	Input
CSE_P3	First ciphertext block address	Input
CSE_P4	First plaintext block address	Output

66.6.4 Decrypt CBC

The DEC_CBC command performs AES-128 decryption in CBC mode on n 128-bit blocks of data with the parameters specified in the table below. The number of blocks parameter is a 32-bit value.

Table 66-19. DEC_CBC command

Register	Value	Data direction
CSE_CMD	0x04	—
CSE_P1	Key ID	Input
CSE_P2	IV address	Input
CSE_P3	Number of blocks (n)	Input
CSE_P4	First ciphertext block address	Input
CSE_P5	First plaintext block address	Output

66.6.5 Generate MAC

The GENERATE_MAC command calculates the MAC of a given message with the parameters specified in the table below. The AES CMAC algorithm is used to calculate a 128-bit MAC output. The message length input is a 64-bit value that specifies the length of the message in bits. A length error (EC = 0x15) is returned if the message length is greater than 0x7fffffff (4 GB).

Table 66-20. GENERATE_MAC command

Register	Value	Data direction
CSE_CMD	0x05	—
CSE_P1	Key ID	Input
CSE_P2	Message length (bits) address	Input
CSE_P3	Message start address	Input
CSE_P4	MAC address	Output

66.6.6 Verify MAC

The VERIFY_MAC command verifies the MAC of a given message with the parameters specified in [Table 66-21](#). The AES CMAC algorithm is used to calculate a 128-bit MAC that is truncated according to the MAC length parameter, which specifies the number of most significant bits in the MAC to compare. A MAC length value of 0 indicates that all 128 bits are compared; a value greater than 128 returns a length error (EC = 0x15). The

message length input is a 64-bit value that specifies the length of the message in bits. A length error (EC = 0x15) is returned if the message length is greater than 0x7fffffff (4 GB). If the input MAC matches the MAC calculated over the message, the CSE_P5 register is set to 0; otherwise, it is set to 1.

Table 66-21. VERIFY_MAC command

Register	Value	Data direction
CSE_CMD	0x06	—
CSE_P1	Key ID	Input
CSE_P2	Message length (bits) address	Input
CSE_P3	Message start address	Input
CSE_P4	MAC address	Input
CSE_P5	MAC length (bits)/ Verification status	Input / Output

66.6.7 Load Key

The LOAD_KEY command updates a memory slot using parameters specified in [Table 66-22](#) according to the SHE memory slot update protocol (see [Secure storage](#)). The 128-bit M1 message contains the UID, Key ID, and Authentication Key ID. The 256-bit M2 message contains the new security flags, the counter, and the key value all encrypted using a derived key generated from the Authentication Key. The 128-bit M3 message is a MAC generated over messages M1 and M2. The 256-bit M4 message is the concatenation of the UID, Key ID, Authorization Key ID, and the encrypted counter value. The 128-bit M5 message is the MAC calculated over message M4.

The derived keys used to generate messages M2, M3, M4, and M5 deviate from the SHE memory slot update protocol when updating the second bank of user keys (KEY_11–KEY_20). During the generation of M2 and M4, K1 and K3 shall be derived using a constant equal to KEY_UPDATE_ENC_C + 0x00800000_00000000_00000000_00000000. Similarly, during generation of M3 and M5, K2 and K4 shall be derived using a constant equal to KEY_UPDATE_MAC_C + 0x00800000_00000000_00000000_00000000. This is intended to prevent aliasing between the two banks of user keys.

Table 66-22. LOAD_KEY command

Register	Value	Data direction
CSE_CMD	0x07	—
CSE_P1	M1 address	Input

Table continues on the next page...

Table 66-22. LOAD_KEY command (continued)

Register	Value	Data direction
CSE_P2	M2 address	Input
CSE_P3	M3 address	Input
CSE_P4	M4 address	Output
CSE_P5	M5 address	Output

66.6.8 Load Plain Key

The LOAD_PLAIN_KEY command updates the RAM key memory slot with a 128-bit plaintext key with the parameter shown in the table below.

Table 66-23. LOAD_PLAIN_KEY command

Register	Value	Data direction
CSE_CMD	0x08	—
CSE_P1	Key address	Input

66.6.9 Export RAM Key

The EXPORT_RAM_KEY command exports the RAM key data with the parameters in the table below. Only keys loaded with the LOAD_PLAIN_KEY command may be exported. The output messages are compatible with the messages used for LOAD_KEY (see [Load Key](#) for details).

Table 66-24. EXPORT_RAM_KEY command

Register	Value	Data direction
CSE_CMD	0x09	—
CSE_P1	M1 address	Output
CSE_P2	M2 address	Output
CSE_P3	M3 address	Output
CSE_P4	M4 address	Output
CSE_P5	M5 address	Output

66.6.10 Initialize RNG

The INIT_RNG command initializes the internal PRNG state with a seed value generated by the TRNG and sets the CSE_SR[RIN] flag. It takes 2048 TRNG clock cycles to generate a seed value. The CSE_CR[DIV] field must be properly configured before this command is executed (see register descriptions for further information). There is a very small probability that this command may return a TRNG error (EC = 0x12) even when the TRNG is operating properly (see [Random number generation](#)). The INIT_RNG command must be called after each reset before the RND command is issued. The command has no parameters as shown the table below.

Table 66-25. INIT_RNG command

Register	Value	Data direction
CSE_CMD	0x0A	—

66.6.11 Extend PRNG Seed

The EXTEND_SEED command extends the state of the PRNG using a 128-bit entropy input value. The current PRNG state and the input data are compressed into a new PRNG state value. This command is issued with the parameter shown in the table below.

Table 66-26. EXTEND_SEED command

Register	Value	Data direction
CSE_CMD	0x0B	—
CSE_P1	Entropy value address	Input

66.6.12 Generate Random Number

The RND command generates a 128-bit random value and updates the state of the internal PRNG. The PRNG state must be initialized after reset using the INIT_RNG command before this command can be issued. This command is issued with the parameter shown in the table below

Table 66-27. RND command

Register	Value	Data direction
CSE_CMD	0x0C	—
CSE_P1	Random value address	Output

66.6.13 Secure Boot

The SECURE_BOOT command loads the command processor firmware and memory slot data from the CSE flash memory blocks, and then it executes the SHE secure boot protocol using the parameters shown in [Table 66-28](#) (see [Secure boot](#) for details). The bootloader size is a 32-bit value. The CSE firmware version is loaded into CSE_P3 register.

The SECURE_BOOT command is issued during the boot processes by the system boot logic. For details about how and when the SECURE_BOOT command is issued on this chip, see the chip-specific CSE information.

Table 66-28. SECURE_BOOT command

Register	Value	Data direction
CSE_CMD	0x0D	—
CSE_P1	Bootloader size (bytes)	Input
CSE_P2	Bootloader start address	Input
CSE_P3	Firmware version	Output

66.6.14 Boot Failure

The BOOT_FAILURE command sets the CSE_SR[BFN] flag and clears the CSE_SR[BOK] flag to disable use of memory slots that have the BOOT_PROT flag set. This command is issued with no parameters as shown in the table below.

Table 66-29. BOOT_FAILURE command

Register	Value	Data direction
CSE_CMD	0x0E	—

66.6.15 Boot OK

The BOOT_OK command sets the CSE_SR[BFN] flag and leaves the CSE_SR[BOK] flag set to confirm successful completion of the secure boot processes. This enables the use of memory slots that have the BOOT_PROT flag set if they are not disabled for some other reason. This command is issued with no parameters as shown in the table below.

Table 66-30. BOOT_OK command

Register	Value	Data direction
CSE_CMD	0x0F	—

66.6.16 Get ID

The GET_ID command returns the UID, CSE_SR[24:31], and a 128-bit MAC calculated over the concatenation of a 128-bit input challenge value, UID, and CSE_SR[24:31]. The MASTER_ECU_KEY is used for the MAC calculation. A value of 0 is returned for the MAC if the MASTER_ECU_KEY slot is empty. The UID output is a 128-bit value with the 8 least significant bits set to 0. This command is issued with the parameters shown in the table below.

Table 66-31. GET_ID command

Register	Value	Data direction
CSE_CMD	0x10	—
CSE_P1	Challenge address	Input
CSE_P2	UID output address	Output
CSE_P3	CSE_SR[24:31] value	Output
CSE_P4	MAC address	Output

66.6.17 Cancel

The CANCEL command aborts processing of the current command and clears the CSE_SR[BSY] flag. This command is issued with no parameters as shown in the table below.

Table 66-32. CANCEL command

Register	Value	Data direction
CSE_CMD	0x11	—

66.6.18 Debug Challenge

The `DEBUG_CHAL` command generates a 128-bit random challenge output value that is used in conjunction with the `DEBUG_AUTH` command. The PRNG state must be initialized after reset using the `INIT_RNG` command before this command can be issued. This command is issued with the parameter shown in the table below.

Table 66-33. `DEBUG_CHAL` command

Register	Value	Data direction
CSE_CMD	0x12	—
CSE_P1	Challenge address	Output

66.6.19 Debug Authorization

The `DEBUG_AUTH` command erases all user keys and sets the `CSE_SR[IDB]` flag which enables internal debugging if the 128-bit authorization input value is valid and no memory slots are write protected. The authorization input is generated using the `DEBUG_CHAL` command output and the UID as described in the *SHE Functional Specification*. If the `DEBUG_CHAL` command is not issued before the `DEBUG_AUTH` command, a command sequence error (`EC = 0x02`) is returned. However, other commands may be issued between the `DEBUG_CHAL` and `DEBUG_AUTH` commands. This command is issued with the parameter shown in the table below.

Table 66-34. `DEBUG_AUTH` command

Register	Value	Data direction
CSE_CMD	0x13	—
CSE_P1	Authorization address	Input

66.6.20 Generate TRNG Random Number

The `TRNG_RND` command generates a 128-bit random output value using the TRNG. It takes 2048 TRNG clock cycles to generate a random value. The `CSE_CR[DIV]` field must be properly configured before this command is executed (see the register descriptions section for further information). This command takes much longer to execute than the `RND` command that should normally be used to generate random values. There is a very small probability that this command may return a TRNG error (`EC = 0x12`) even when the TRNG is operating properly (see [Random number generation](#)). This command is issued with the parameter shown in the table below.

Table 66-35. TRNG_RND Command

Register	Value	Data direction
CSE_CMD	0x14	—
CSE_P1	Random value address	Output

66.6.21 Initialize CSE

The INIT_CSE command loads the command processor firmware and memory slot data from the CSE flash memory blocks into local memory. It does not execute the secure boot protocol. The CSE firmware version is loaded into the CSE_P1 register as shown in the table below. This command must be issued before any other command when secure boot is not enabled.

Table 66-36. INIT_CSE Command

Register	Value	Data direction
CSE_CMD	0x15	—
CSE_P1	Firmware version	Output

66.6.22 Miyaguchi-Preneel (MP) Compression

The MP_COMPRESS command is a one-way compression function used to derive a 128-bit output from a given message with the parameters specified in [Table 66-37](#). The AES ECB algorithm is used to derive the 128-bit output as described in the SHE specification. The message length input is a 64-bit value which specifies the length of the message in bits. A length error (EC=0x15) is returned if the message length is greater than 0x7fffffff (4GB).

Table 66-37. MP_COMPRESS Command

Register	Value	Data direction
CSE_CMD	0x016	
CSE_P1	Message length (bits) address	Input
CSE_P2	Message start address	Input
CSE_P3	Output address	Output

Chapter 67

Password and Device Security Module (PASS)

67.1 Chip-specific PASS information

This chip has one instance of the PASS module. PASS implements password-based read and write protection for flash memory blocks.

This chip implements four protection regions.

67.1.1 Accesses to unimplemented PASS registers

On this chip, accesses to unimplemented PASS registers generate bus aborts.

67.1.2 PASS_LOCKm_PGn register bit mapping

Each password group defined in the DCF records is associated with a set of four PASS_LOCKm_PGn registers. Each implemented bit in these registers maps to a specific memory block.

NOTE

These registers are initialized via DCF records, so the reset values depend on user settings.

67.1.2.1 PASS_LOCK0_PGn register bit mapping

The flash memory blocks mapped to the [Password Group n - Lock 0 Status Register \(PASS_LOCK0_PGn\)](#) bits, and the corresponding PASS protection regions, are as follows.

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31

Table continues on the next page...

Protection Region	Flash Memory Block Name	Register Field	
		TSLOCK	Reserved
PASS Read Lock Protection Region 0	UTest NVM Block Space 16 KB	LOWLOCK[13:0]	
—	—		
—	—		
—	—		
—	—		
—	—		
—	—		
—	—		
—	—		
—	—		
—	—		
—	—		
—	—		
—	—		
PASS Read Lock Protection Region 2	64 KB EEPROM Data Block1 - Partition 1	MIDLOCK[15:0]	
PASS Read Lock Protection Region 2	64 KB EEPROM Data Block0 - Partition 0		
—	—		
—	—		
—	—		
—	—		
—	—		
—	—		
—	—		
—	—		
—	—		
—	—		
—	—		
—	—		
PASS Read Lock Protection Region 2	64 KB EEPROM Data Block1 - Partition 3	MIDLOCK[15:0]	
PASS Read Lock Protection Region 2	64 KB EEPROM Data Block0 - Partition 2		
—	—		
—	—		
—	—		
—	—		
—	—		
—	—		
—	—		
—	—		
—	—		
—	—		
—	—		
—	—		
—	—		

67.1.2.2 PASS_LOCK1_PG n register bit mapping

The flash memory blocks mapped to the [Password Group \$n\$ - Lock 1 Status Register \(PASS_LOCK1_PG \$n\$ \)](#) bits, and the corresponding PASS protection region, are as follows.

NOTE

This register cannot be used to manage the blocks mapped to the register's implemented bits because the CSE module has exclusive access to these blocks. The bits' value is 1 to reflect

the corresponding blocks' locked state, and the bits cannot be written to change that state.

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
Register Field	Reserved															HIGHLOCK[15:0]																
Protection Region	Flash Memory Block Name																															
—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	
—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—
—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—
—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—
—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—
—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—
—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—
—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—
—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—
—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—
—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—
—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—
—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—
—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—
—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—
—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—
—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—
—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—
—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—
—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—
—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—
—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—
—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—
—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—
—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—
—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—
—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—
—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—
—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—
—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—
—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—
—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—
—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—
—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—
—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—									

67.1.2.3 PASS_LOCK2_PGn register bit mapping

The flash memory blocks mapped to the Password Group n - Lock 2 Status Register (PASS_LOCK2_PG n) bits, and the corresponding PASS protection region, are as follows.

Chip-specific PASS information

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
Register Field	L_256LCK[31:0]																															
Flash Memory Block Name	256 KB Application Code Block31 - Partition 9	256 KB Application Code Block30 - Partition 9	256 KB Application Code Block29 - Partition 9	256 KB Application Code Block28 - Partition 9	256 KB Application Code Block27 - Partition 9	256 KB Application Code Block26 - Partition 9	256 KB Application Code Block25 - Partition 9	256 KB Application Code Block24 - Partition 9	256 KB Application Code Block23 - Partition 8	256 KB Application Code Block22 - Partition 8	256 KB Application Code Block21 - Partition 8	256 KB Application Code Block20 - Partition 8	256 KB Application Code Block19 - Partition 8	256 KB Application Code Block18 - Partition 8	256 KB Application Code Block17 - Partition 8	256 KB Application Code Block16 - Partition 8	256 KB Application Code Block15 - Partition 7	256 KB Application Code Block14 - Partition 7	256 KB Application Code Block13 - Partition 7	256 KB Application Code Block12 - Partition 7	256 KB Application Code Block11 - Partition 7	256 KB Application Code Block10 - Partition 7	256 KB Application Code Block9 - Partition 7	256 KB Application Code Block8 - Partition 7	256 KB Application Code Block7 - Partition 6	256 KB Application Code Block6 - Partition 6	256 KB Application Code Block5 - Partition 6	Calibration Block4 - Partition 6	Calibration Block3 - Partition 6	Calibration Block2 - Partition 6	Calibration Block1 - Partition 6	Boot Code Block0 - Partition 6
Protection Region	PASS Read Lock Protection Region 1																															

67.1.2.4 PASS_LOCK3_PGn register bit mapping

The flash memory blocks mapped to the [Password Group *n* - Lock 3 Status Register \(PASS_LOCK3_PGn\)](#) bits, and the corresponding PASS protection region, are as follows.

Reserved bits read as 0.

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31

Table continues on the next page...

[illegible]

67.1.3 PASS chip select (client select)

On this chip, the chip select (client select) for the PASS module is CS[3].

For details about CS[3], see the DCF Records chapter.

67.1.4 Flash memory read protection scenarios

The following table summarizes the scenarios of read lock (RL) region protection on this chip. For more information, see [Flash Memory Read Protection](#).

Table 67-1. Flash memory read protection truth table

Inputs				Outputs	
Life Cycle	Censorship DCF client	Debug Interface	LOCK3[RLx]	Read Protected RL[0] (UTest)	Read Protected RL[3:1] (other regions) ^{1, 2}
Customer Delivery	Don't care	Don't care	Don't care	Readable	Readable

Table continues on the next page...

Table 67-1. Flash memory read protection truth table (continued)

Inputs				Outputs	
Life Cycle	Censorship DCF client	Debug Interface	LOCK3[RLx]	Read Protected RL[0] (UTest)	Read Protected RL[3:1] (other regions) ^{1, 2}
OEM Production or In Field	0x55AA ³	Don't care	Don't care	Readable	Readable
	Any other value	Blocked	Don't care	Readable	Readable
		Enabled	Locked	Read Blocked ⁴	Read Blocked ⁴
			Unlocked	Readable	Readable
Failure Analysis	0x55AA	Don't care	Don't care	Readable	Readable
	Any other value	Don't care	Locked	Readable	Read Blocked
			Unlocked	Readable	Readable

1. When Life Cycle is other than In Field: If the BAM cannot find a valid RCHW because the corresponding regions are Read Blocked, the boot mode is switched to serial.
2. After the Life Cycle state OEM Production, the UTest space reserved for passwords is protected by the flash memory interface instead of PASS. PASS has no effect on the space.
3. This condition is also valid if the JTAG password is used to break the censorship scheme.
4. When Life Cycle is In Field, the BAM will not switch to serial boot mode. Instead, it will keep searching for a valid RCHW until SWT requests a reset.

67.1.5 Debug Interface Access

Whether the Debug Interface Access is enabled or blocked depends on multiple input conditions. The following table shows the the Debug Interface Access scenarios on this chip.

Table 67-2. Debug Interface Access truth table

Life Cycle	Production Disable	Censorship DCF client	Debug Enable LOCK3[DBL] bit	JTAG Password	Debug Interface Access status
Customer Delivery	Don't care	Don't care	Don't care	Don't care	Enabled
OEM Production	Don't care	0x55AA	Don't care	Don't care	Enabled
		Any other value	Unlocked (DBL = 0)	Don't care	Enabled
			Locked (DBL = 1)	Matched	Enabled
				Not Matched	Blocked
In Field	Off	0x55AA	Don't care	Don't care	Enabled
		Any other value	Unlocked (DBL = 0)	Don't care	Enabled
			Locked (DBL = 1)	Matched	Enabled
				Not Matched	Blocked
	On	Don't care	Don't care	Don't care	Blocked
Failure Analysis ¹	Don't care	0x55AA	Don't care	Don't care	Enabled
		Any other value	Unlocked (DBL = 0)	Don't care	Enabled
			Locked (DBL = 1)	Matched	Enabled
				Not Matched	Blocked

1. When advancing the Life Cycle to FA, program the Censorship DCF client with 0x55AA to enable the Debug Interface Access for Freescale.

When the Life Cycle is Customer Delivery, the Debug Interface Access is fully enabled regardless of any boundary conditions.

When the Life Cycle is OEM Production, In Field, or Failure Analysis, the Debug Interface Access depends on the Censorship DCF client settings, the 256-bit JTAG Password entered via JTAG mechanism, and the LOCK3[DBL] bit.

For the Life Cycle of In Field, the Debug Interface is further conditioned by the Production Disable feature. If the Production Disable feature is enabled when the Life Cycle is In Field, the Debug Interface Access may be permanently blocked because Production Disable overrides all other debug controls. The preceding table emphasizes this scenario by shading the set of conditions and the result in grey.

67.1.6 Duration of Debug Interface Access via JTAG password

[JTAG Password Challenge](#) explains the method for enabling Debug Interface Access by writing the [JTAG_PASSWORD register](#) via an external tool. Debug Interface Access granted by a password match remains enabled until the next "destructive" reset. On this chip, a "destructive" reset is any of the following:

- POR
- A reset caused by:
 - LVD or HVD event
 - Temperature threshold detection event
 - RESET pin
 - STCU2 self test
 - FOSU timeout

67.1.7 Production Disable enablement

The enablement of the Production Disable feature is done via a DCF client that is not accessible by software and that is automatically loaded during the reset phase of the device. For more information, see [Production disable](#).

CAUTION

Be careful when configuring the Production Disable feature because enabling Production Disable might permanently block the Debug Interface Access.

67.1.8 PASS_LCSTAT[LIFE] value source

On this chip, the SSCM is the "external" module from which PASS_LCSTAT[LIFE] derives its reset value. The value of PASS_LCSTAT[LIFE] directly reflects the value of SSCM_LCSTAT[LC].

67.2 Introduction**67.2.1 Overview**

The Password and device security (PASS) module shown in the following figure, receives password challenges and determines their validity. It also maintains the device security and access states.

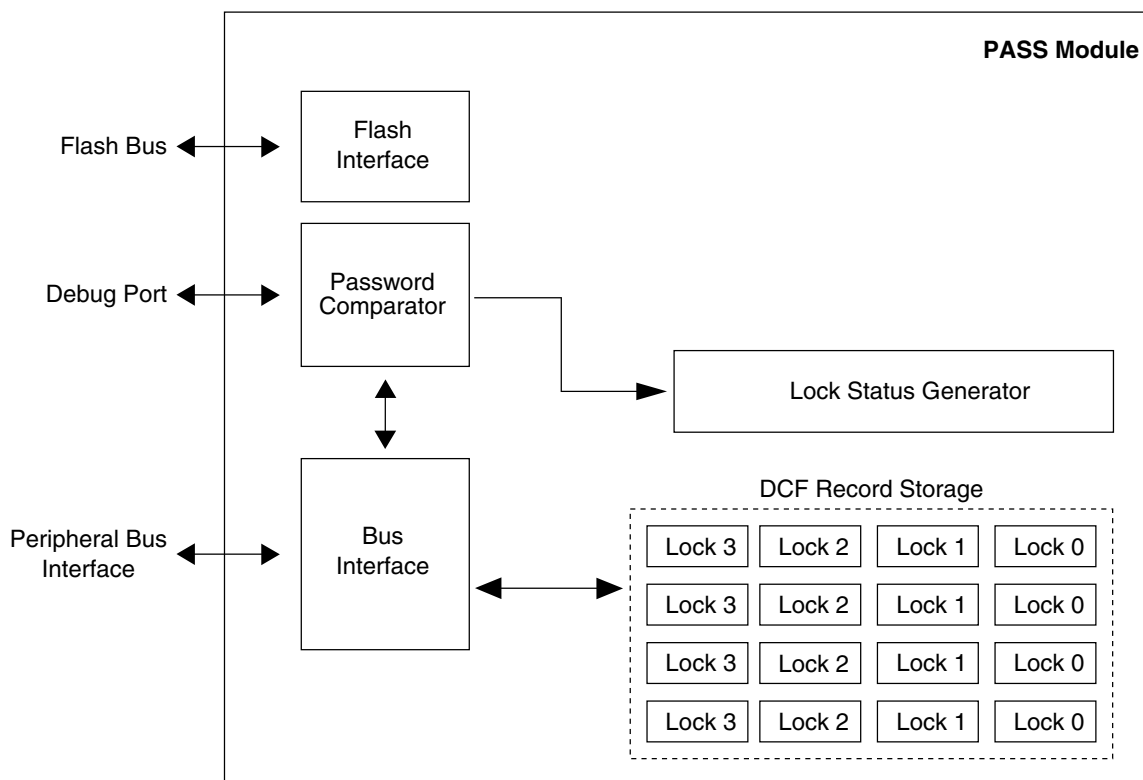


Figure 67-1. PASS module block diagram

67.2.2 Features

The PASS includes these distinctive features:

- Life Cycle status
- Password comparison
- JTAG password comparison
- Production Disable control
- Flash read, write, and erase control

67.2.3 Modes of operation

The PASS operates identically in all system modes.

67.3 Memory map and register definition

This section provides a detailed description of all memory-mapped registers and DCF clients in the PASS.

Where a DCF client is listed, the reset value of that register may automatically be pre-loaded during reset based on the data stored in DCF Records in flash UTEST region.

The following table shows the memory map for the PASS. Note that all addresses are offsets; the absolute address for registers may be calculated by adding the specified offset to the base address of the PASS. The address of the DCF client is local to the PASS module and must be qualified with the appropriate Client Select (CS[14:0]) for the PASS module, as defined in the DCF Record Section of the Reference Manual.

NOTE

Non-implemented registers generate bus aborts if enabled.

The following registers are available in the PASS. The shaded bits are reserved for future use. For future compatibility, only write '0' to these bits; they always read '0'.

PASS memory map

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
0	Life Cycle Status Register (PASS_LCSTAT)	32	R	See section	67.3.1/3303

Table continues on the next page...

PASS memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
8	Challenge Selector Register (PASS_CHSEL)	32	R/W	0000_0000h	67.3.2/3304
10	Challenge Status Register (PASS_CSTAT)	32	R	0000_0000h	67.3.3/3305
20	Challenge Input Register (PASS_CIN0)	32	W (always reads 0)	0000_0000h	67.3.4/3305
24	Challenge Input Register (PASS_CIN1)	32	W (always reads 0)	0000_0000h	67.3.4/3305
28	Challenge Input Register (PASS_CIN2)	32	W (always reads 0)	0000_0000h	67.3.4/3305
2C	Challenge Input Register (PASS_CIN3)	32	W (always reads 0)	0000_0000h	67.3.4/3305
30	Challenge Input Register (PASS_CIN4)	32	W (always reads 0)	0000_0000h	67.3.4/3305
34	Challenge Input Register (PASS_CIN5)	32	W (always reads 0)	0000_0000h	67.3.4/3305
38	Challenge Input Register (PASS_CIN6)	32	W (always reads 0)	0000_0000h	67.3.4/3305
3C	Challenge Input Register (PASS_CIN7)	32	W (always reads 0)	0000_0000h	67.3.4/3305
100	Password Group n - Lock 0 Status Register (PASS_LOCK0_PG0)	32	R/W	See section	67.3.5/3306
104	Password Group n - Lock 1 Status Register (PASS_LOCK1_PG0)	32	R/W	See section	67.3.6/3308
108	Password Group n - Lock 2 Status Register (PASS_LOCK2_PG0)	32	R/W	See section	67.3.7/3309
10C	Password Group n - Lock 3 Status Register (PASS_LOCK3_PG0)	32	R/W	See section	67.3.8/3310
110	Password Group n - Lock 0 Status Register (PASS_LOCK0_PG1)	32	R/W	See section	67.3.5/3306
114	Password Group n - Lock 1 Status Register (PASS_LOCK1_PG1)	32	R/W	See section	67.3.6/3308
118	Password Group n - Lock 2 Status Register (PASS_LOCK2_PG1)	32	R/W	See section	67.3.7/3309
11C	Password Group n - Lock 3 Status Register (PASS_LOCK3_PG1)	32	R/W	See section	67.3.8/3310
120	Password Group n - Lock 0 Status Register (PASS_LOCK0_PG2)	32	R/W	See section	67.3.5/3306
124	Password Group n - Lock 1 Status Register (PASS_LOCK1_PG2)	32	R/W	See section	67.3.6/3308

Table continues on the next page...

PASS memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
128	Password Group n - Lock 2 Status Register (PASS_LOCK2_PG2)	32	R/W	See section	67.3.7/3309
12C	Password Group n - Lock 3 Status Register (PASS_LOCK3_PG2)	32	R/W	See section	67.3.8/3310
130	Password Group n - Lock 0 Status Register (PASS_LOCK0_PG3)	32	R/W	See section	67.3.5/3306
134	Password Group n - Lock 1 Status Register (PASS_LOCK1_PG3)	32	R/W	See section	67.3.6/3308
138	Password Group n - Lock 2 Status Register (PASS_LOCK2_PG3)	32	R/W	See section	67.3.7/3309
13C	Password Group n - Lock 3 Status Register (PASS_LOCK3_PG3)	32	R/W	See section	67.3.8/3310

67.3.1 Life Cycle Status Register (PASS_LCSTAT)

The Life Cycle Status Register reflects the production status and DCF Censorship client status of the device. It is possible to mature the device (move the device life cycle forward), but never to revert the life cycle to an earlier state.

The reset value of the LCSTAT register LIFE field derives from an external life cycle management module (like SSCM). It is a 3-bit life cycle. This register simply displays the life cycle input from external blocks.

Address: 0h base + 0h offset = 0h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	CNS	JUN	1	0												
W																
Reset	*	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0												LIFE			
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	*	*	*

* Notes:

- LIFE field: Reset value is the Life Cycle of the device, and is unaffected by reset.
- CNS field: Reset value depends on the Censorship DCF client

PASS_LCSTAT field descriptions

Field	Description
0 CNS	This field indicates whether the DCF Censorship client has been written with a DCF record to 0x55AA or if the DCF Censorship client contains any other value than 0x55AA.

Table continues on the next page...

PASS_LCSTAT field descriptions (continued)

Field	Description
	The bit can be set (or cleared) via DCF Record in UTest Flash. NOTE: If no DCF record has been written to affect the value of the CNS bit, its reset value will be 1. 1 DCF Censorship Client value is anything other than 0x55AA. 0 DCF Censorship client value is 0x55AA.
1 JUN	This field indicates whether the device has been temporarily uncensored. Successful transmission of the JTAG password sets this bit to '1', otherwise it will be 0. (In life cycle FA, the JUN bit will always be 0.) Unsuccessful transmission clears this bit.
2 Reserved	This field is reserved. This read-only field is reserved and always has the value 1.
3–28 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
29–31 LIFE	The life cycle setting is manipulated externally (SSCM). Pass displays a 3-bit input life cycle 000 Failure Analysis 001 Reserved 010 OEM Production 011 Customer Delivery 100 Reserved 101 Reserved 110 MCU Production 111 In Field

67.3.2 Challenge Selector Register (PASS_CHSEL)

This register determines which password group is challenged via the Challenge Input Register.

When the Master Only (MO) bit of the PASS_LOCK3_PGn register is set, only the master that unlocked access to the passgroup settings will be able to access these registers. The CMST field of the Challenge Status Register will reflect the locking master. When the Master Only (MO) bit of the PASS_LOCK3_PGn register is not set, the passgroup settings can be accessed by any master.

When a master writes this register, its ID sets the CMST field of the Challenge Status register.

Address: 0h base + 8h offset = 8h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0																GRP															
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	

PASS_CHSEL field descriptions

Field	Description
0–29 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
30–31 GRP	Password Group to unlock 00 Password Group 0 01 Password Group 1 10 Password Group 2 11 Password Group 3

67.3.3 Challenge Status Register (PASS_CSTAT)

This register provides status information for the password challenge.

Only the master shown in this register can access the CINn registers - access by other masters result in a comparison fail regardless of the values written.

Address: 0h base + 10h offset = 10h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0																											CMST				
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	

PASS_CSTAT field descriptions

Field	Description
0–27 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
28–31 CMST	ID of the master which has last written the CHSEL register

67.3.4 Challenge Input Register (PASS_CINn)

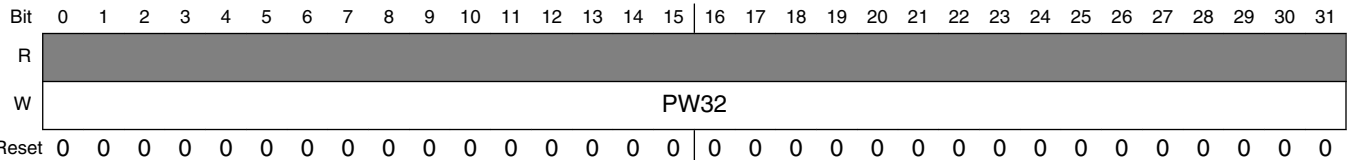
This register is used to provide the Challenge word *n* for a password group as selected by the Challenge Selector Register.

After programming the Challenge Selector Register, the password challenge needs to be written with eight 32-bit writes from CIN0 to CIN7 in that order. The most significant 32 bits of the password challenge must be written to CIN0. Once CIN7 is written, the password is compared. The write access only completes after the comparison has been

executed. In case of a successful password comparison, the PASS module stores the ID of the master which has written CIN0 to CIN7 in the PASS_LOCK3_PGn register of the selected password group, as well as clears the PGL bit of that register.

All accesses to these registers need to be made by the bus master that wrote the CHSEL register previously. If another master writes any of the CIN registers before the original master has written CIN7, the comparison fails.

Address: 0h base + 20h offset + (4d × i), where i=0d to 7d



PASS_CINn field descriptions

Field	Description
0–31 PW32	32 bits of the 256-bit password challenge. Only 32-bit writes may be used.

67.3.5 Password Group n - Lock 0 Status Register (PASS_LOCK0_PGn)

Each Password Group has one Lock 0 Status register. This register shows the current lock state for the Flash blocks. Each time a password group is unlocked the register becomes writable. For additional details of the LOCK bits, refer to the Flash memory description.

The LOCK bits do not come into effect before the life cycle has matured to *OEM Production* or older.

LOCK0_PGn register functions are as shown in this section.

NOTE

Lock registers can only be accessed with 32-bit operations.

The initial value of the lock bits is set via DCF records stored in UTEST Flash block and copied over during functional reset. The default value before any DCF record is written is '1', indicating that a block is locked.

During the Customer Delivery life cycle, the user must program a valid password or a DCF record that writes to this register to unprotect the UTEST for each password group in order to prevent permanent and irreversible locking of Flash blocks.

Warning

Failure to program valid passwords or DCF records for all password groups before the life cycle matures to OEM Production renders the part unusable.

Once a password group is unlocked, software can set or clear any of the lock bits in the group, by writing to the registers.

The resulting lock status of a Flash block is determined by the combination of the lock bits of all password groups. If a block is locked in multiple groups, then all lock bits for the block need to be cleared before program and erase is possible. However, unless the life cycle is OEM Production or older, the Flash blocks are always unlocked, regardless of the values in the password groups.

Address: 0h base + 100h offset + (16d × i), where i=0d to 3d

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	TSLOCK	Reserved	LOWLOCK													
W																
Reset	*		*	*	*	*	*	*	*	*	*	*	*	*	*	*
Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	MIDLOCK															
W																
Reset	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*

* Notes:

- MIDLOCK field: For each implemented bit of this field, if no DCF records have been written to change the lock state, the bit will reset to '1'.
- LOWLOCK field: For each implemented bit of this field, if no DCF records have been written to change the lock state, the bit will reset to '1'.
- TSLOCK field: If no DCF records have been written to change the TS bit state, the bit will reset to '1'.

PASS_LOCK0_PGn field descriptions

Field	Description
0 TSLOCK	UTest NVM Lock. This bit is used to lock the UTest NVM block from programs (erase protection not needed since UTest NVM is OTP and not erasable).
1 Reserved	This field is reserved.
2–15 LOWLOCK	Low Block Lock. A value of 1 in a bit of the lock register signifies that the corresponding block is locked for program and erase.
16–31 MIDLOCK	Mid Block Lock. A value of 1 in a bit of the lock register signifies that the corresponding block is locked for program and erase.

67.3.6 Password Group *n* - Lock 1 Status Register (PASS_LOCK1_PG*n*)

Each Password Group has one Lock 1 Status Register. This register shows the current lock state for the Flash blocks. Each time a password group is unlocked, the register becomes writable . For additional details of the LOCK bits, refer to the Flash description.

The LOCK bits do not come into effect before the life cycle matures to OEM Production or older.

LOCK1_PG*n* register functions are as shown in this section.

NOTE

Lock registers can only be accessed with 32-bit operations.

The initial value of the lock bits is set via DCF records stored in UTEST Flash block and copied over during functional reset . The default value before any DCF record is written is '1', which indicates a block is locked. If no DCF records have been written to change the lock state, the bits will reset to '1'.

During the Customer Delivery life cycle, the user must program a valid password or a DCF record that writes this register to unprotect the UTEST for each password group to prevent permanent and irreversible locking of Flash blocks.

CAUTION

Failure to program valid passwords or DCF records for all password groups before the life cycle matures to OEM Production renders the part unusable. However, additional DCF Records can be written to UTEST to change the initial value of this register.

Once a password group is unlocked, software can set or clear any of the lock bits in the group by writing to the registers.

The resulting lock status of a Flash block is determined by the combination of the lock bits of all password groups. If a block is locked in multiple groups, then all lock bits for the block need to be cleared before program and erase is possible. However, unless the life cycle is older than OEM Production, the Flash blocks are always unlocked, regardless of the values in the password groups.

Address: 0h base + 104h offset + (16d × i), where i=0d to 3d

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0																HIGHLOCK															
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*

* Notes:

- HIGHLOCK field: For each implemented bit of this field, if no DCF records have been written to change the lock state, the bit will reset to '1'.

PASS_LOCK1_PG n field descriptions

Field	Description
0–15 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
16–31 HIGHLOCK	High Block Lock. A value of 1 in a bit of the lock register signifies that the corresponding block is locked for program and erase.

67.3.7 Password Group n - Lock 2 Status Register (PASS_LOCK2_PG n)

Each Password Group has one Lock 2 Status Register. This register shows the current lock state for the Flash blocks. Each time a password group is unlocked, the register becomes writable. For additional details of the LOCK bits, refer to the Flash description.

The LOCK bits do not come into effect before the life cycle is matured to OEM Production or older.

NOTE

Lock registers can only be accessed with 32-bit operations.

LOCK2_PG n register functions are as shown in this section.

The initial value of the lock bits is set via DCF records stored in UTEST Flash block and copied over during functional reset. The default value before any DCF record is written is '1', which indicates a block is locked.

During the Customer Delivery life cycle, the user must program a valid password or a DCF record that writes this register to unprotect the UTEST for each password group in order to prevent permanent and irreversible locking of Flash blocks.

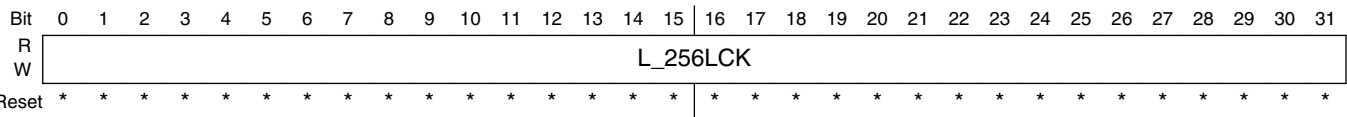
Warning

Failure to program valid passwords or DCF records for all password groups before the life cycle matures to OEM Production renders the part unusable. However, additional DCF Records can be written to UTEST to change the initial value of this register.

Once a password group is unlocked, software can set or clear any of the lock bits in the group, by writing to the registers.

The resulting lock status of a Flash block is determined by the combination of the lock bits of all password groups. If a block is locked in multiple groups, then all lock bits for the block need to be cleared before program and erase is possible. However, unless the life cycle is older than OEM Production, the Flash blocks are always unlocked, regardless of the values in the password groups.

Address: 0h base + 108h offset + (16d × i), where i=0d to 3d



- * Notes:
- L_256LCK field: For each implemented bit of this field, if no DCF records have been written to change the lock state, the bit resets to '1'.

PASS_LOCK2_PGn field descriptions

Field	Description
0–31 L_256LCK	Lower 256KByte Block Lock. A value of 1 in a bit of the lock register signifies that the corresponding block is locked for program and erase.

67.3.8 Password Group n - Lock 3 Status Register (PASS_LOCK3_PGn)

Each Password Group has one Lock 3 Status Register. This register shows the current lock state for the Flash blocks. Each time a password group is unlocked, the register becomes writable.

The LOCK bits do not come into effect before the life cycle matures to OEM Production or older.

LOCK3_PGn register functions are as shown in this section.

NOTE

Lock registers can only be accessed with 32-bit operations.

The initial value of the lock bits is set via DCF records stored in UTEST Flash block and copied over during functional reset. The default value before any DCF record is written is '1', which indicates a block is locked.

During the Customer Delivery life cycle, the user must program a valid password or a DCF record that writes this register to unprotect the UTEST for each password group in order to prevent permanent and irreversible locking of Flash blocks.

Warning

Failure to program valid passwords or DCF records for all password groups before the life cycle matures to OEM Production renders the part unusable. However, additional DCF Records can be written to UTEST to change the initial value of this register.

Once a password group is unlocked, software can set or clear any of the lock bits in the group, by writing to the registers.

The resulting lock status of a Flash block is determined by the combination of the lock bits of all password groups. If a block is locked in multiple groups, then all lock bits for the block need to be cleared before program and erase is possible. However, unless the life cycle is older than OEM Production, the Flash blocks are always unlocked, regardless of the values in the password groups.

Once a censored device enters the Failure Analysis life cycle, read protection is active regardless whether the system is in debug mode or not. It is still possible to unlock Flash regions with the correct passwords, however. Flash Utest region controlled by LOCK3[0] is read unlocked in Failure Analysis life cycle.

NOTE

Do not toggle DBL and RLx bits with the same write command. Use one instruction to program the DBL bit and a separate instruction to toggle the RLx bits.

Address: 0h base + 10Ch offset + (16d × i), where i=0d to 3d

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	PGL	DBL	MO	0	MSTR				0			0	RL3	RL2	RL1	RL0
W																
Reset	*	*	*	0	1	1	1	1	0	0	0	*	*	*	*	*

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	U_256LCK															
W																
Reset	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*

* Notes:

- U_256LCK field: For each implemented bit of this field, if no DCF records have been written to change the lock state, the bit will reset to '1'.
- RL0 field: If no DCF records have been written to change the RL0 bit lock state, the bit will reset to a logic 1. This Flash block region is always readable if the life cycle is Customer Delivery or earlier. To determine if this region is readable for Life Cycle states later than Customer Delivery, consult the Flash Memory Read Protection Truth Table.
- RL1 field: If no DCF records have been written to change the RL1 bit lock state, the bit will reset to a logic 1. This Flash block region is always readable if the life cycle is Customer Delivery or earlier. To determine if this region is readable for Life Cycle states later than Customer Delivery, consult the Flash Memory Read Protection Truth Table.

Memory map and register definition

- RL2 field: If no DCF records have been written to change the RL2 bit lock state, the bit will reset to a logic 1. This Flash block region is always readable if the life cycle is Customer Delivery or earlier. To determine if this region is readable for Life Cycle states later than Customer Delivery, consult the Flash Memory Read Protection Truth Table.
- RL3 field: If no DCF records have been written to change the RL3 bit lock state, the bit will reset to a logic 1. This Flash block region is always readable if the life cycle is Customer Delivery or earlier. To determine if this region is readable for Life Cycle states later than Customer Delivery, consult the Flash Memory Read Protection Truth Table.
- Reserved field: The reset value is chip-specific. See the chip-specific PASS information
- MO field: If no DCF records have been written to change the MO bit lock state, the bit will reset to '1'.
- DBL field: If no DCF records have been written to change the DBL bit lock state, the bit will reset to '1'.
- PGL field: Reset value depends on life cycle. If life cycle is Customer Delivery or earlier, the value will be 0, otherwise 1.

PASS_LOCK3_PGn field descriptions

Field	Description
0 PGL	Password Group Lock. This bit may be set in software to lock the password group, when the device is older than Customer Delivery. Before that the value of the bit is always 0. This bit is only updatable from SW path. It cannot be updated from DCF load from Flash. This bit would be automatically locked once Flash Scanning is finished after functional reset. This bit can be cleared by writing a valid password to the Password Challenge Input Registers. 0 Password group registers are unlocked. All four registers in the Password group may be read and written without restriction. 1 Password group registers are locked. Writes to the password group registers have no effect. Read accesses work normally. (If life cycle is Customer Delivery or earlier, this bit has no effect and the registers are always unlocked.)
1 DBL	Debug Interface Lock. The default value of this bit would be '1' if not updated from flash. At power on, this bit would be updated from DCF records in flash. This bit cannot be updated from FLASH after first functional reset. After first functional reset, it can only be updated by writing to it once the passgroup has been unlocked via the PASS Password challenge mechanism. 0 Debug interface is unlocked. Accessed to all JTAG clients is allowed 1 Debug interface is locked. Only access to JTAG password challenge register is possible. Access to all other JTAG clients is blocked
2 MO	Master Only. 0 All masters can modify the passgroup settings if PGL is cleared 1 Only the Master which unlocked the passgroup can change the passgroup settings
3 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
4–7 MSTR	Master Access. Only the Master with this matching master ID can change the passgroup settings. This field is loaded with the ID of the master which unlocked the passgroup once the correct password is provided.
8–10 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
11 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

Table continues on the next page...

PASS_LOCK3_PGn field descriptions (continued)

Field	Description
12 RL3	Region 3 is protected. The default value of this bit would be '1' if not updated from flash. At power on, this bit would be updated from records loaded in flash. This bit cannot be updated from FLASH after first functional reset. After first functional reset, It can only be updated by Test Access Port or Through PASS Password challenge mechanism. 0 Region 3 is not protected. 1 Region 3 is protected.
13 RL2	Region 2 is protected. The default value of this bit would be '1' if not updated from flash. At power on, this bit would be updated from records loaded in flash. This bit cannot be updated from FLASH after first functional reset. After first functional reset, It can only be updated by Test Access Port or Through PASS Password challenge mechanism. 0 Region 2 is not protected. 1 Region 2 is protected.
14 RL1	Region 1 is protected. The default value of this bit would be '1' if not updated from flash. At power on, this bit would be updated from records loaded in flash. This bit cannot be updated from FLASH after first functional reset. After first functional reset, It can only be updated by Test Access Port or Through PASS Password challenge mechanism. 0 Region 1 is not protected. 1 Region 1 is protected.
15 RL0	Region 0 is protected. The default value of this bit would be '1' if not updated from flash. At power on, this bit would be updated from records loaded in flash. This bit cannot be updated from FLASH after first functional reset. After first functional reset, It can only be updated by Test Access Port or Through PASS Password challenge mechanism. 0 Region 0 is not protected. 1 Region 0 is protected.
16–31 U_256LCK	Upper 256 KB Block Lock. A value of 1 in a bit of the lock register signifies that the corresponding block is locked for program and erase.

67.4 DCF clients

The following table shows the DCF clients for the PASS.

Where a DCF client is listed, the reset value of that register may automatically be pre-loaded during reset based on the data stored in DCF Records in Flash UTEST region.

The address of the DCF client is local to the PASS module and must be qualified with the appropriate Client Select (CS[14:0]) for the PASS module, as defined in the DCF Record Section of the Reference Manual or in DCF sheet attached to this document.

Table 67-36. PASS DCF clients

Offset	Register	Access	Reset Value	DCF Client Address (ADDR[14:0])	Section
Censorship					
None	Censorship	None	0x0000_0000	0x00B0	Censorship
Production Disable					
None	Production Disable	None	0x0000_0000	0x00C0	Production Disable

67.4.1 Censorship

Censorship is a DCF client that is *not* accessible by software. It is automatically pre-loaded during reset with data stored in DCF Record format in the Flash UTEST region.

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R																
W																
Reset	x ¹	x ¹	x ¹	x ¹	x ¹	x ¹	x ¹	x ¹	x ¹	x ¹	x ¹	x ¹	x ¹	x ¹	x ¹	x ¹

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R																
W	x ²	x ²	x ²	x ²	x ²	x ²	x ²	x ²	x ²	x ²	x ²	x ²	x ²	x ²	x ²	x ²
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

¹ This bit is not defined

² This value is loaded by DCF Record during reset

Field	Description
16:31	Censorship If this value is 0x55AA the MCU is NOT censored If this value is anything other than 0x55AA, the MCU is censored

67.4.2 Production Disable

Production Disable is a DCF client that is not accessible by software. It is automatically pre-loaded during reset with data stored in DCF Record format in the Flash UTEST region. The Production disable DCF client defines the Debug Disable aspects of the Production Disable feature.

The DCF client implements 1 bit with a default value of '0'. The DCF client can only be written once and writes from subsequent DCF records are ignored.

For more information on Production disable, see [Production disable](#)

DCF Client Address: 0x00C0 Production Disable

Access: DCF Record Write Once Only

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R																
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R																
W																DD ¹
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

¹ This value is loaded by DCF Record during reset

Figure 67-34. Production Disable DCF client

Table 67-37. Production Disable DCF Client field descriptions

Field	Description
0-30	This bit is reserved and always has the value 0.
31	Debug disable.
DD	'0' = Debug Interface is Enabled and Disabled using Censorship, JTAG password and PASS_LOCK3[DBL] bit (see chip-specific PASS information). '1' = Debug Interface is disabled

67.4.2.1 Debug Disable (DD)

Debug Disable is an optional feature that, once the life cycle is In Field or beyond, overrides all other debug controls and disables the debug interface. The Debug Disable feature has a higher priority than Censorship, JTAG password and the PASS LOCK3[DBL] bit.

67.5 Functional description

The primary purpose of the PASS is to provide fine-grained access control to the Flash and debugger interface. It utilizes the DCF mechanism, as described in the SSCM section.

67.5.1 Censorship

Censorship is the overall control that enables and disables two independent security features: Debug Interface access and flash memory read protection.

The Debug Interface access is controlled by the CNS and LIFE bits of the LCSTAT register ([Life Cycle Status Register \(PASS_LCSTAT\)](#)) according to the Debug Interface Access truth table. The flash memory read protection is also controlled by the CNS and LIFE bits of the LCSTAT register according to the Flash memory read protection truth table.

The CNS bit is set or cleared by executing a DCF record that writes to the Censorship DCF client (See [Censorship](#)). The LIFE bits of the LCSTAT register are controlled by the SSCM using user programmed values from specific locations in the UTEST flash.

67.5.2 Debug Interface Access

The debug interface is fully enabled in Customer Delivery life cycle. But from OEM Production life cycle, it can be blocked using censorship and unlocked by either entering a 256 bits password via JTAG or by software clearing the DBL bit the PASS_LOCK3 registers (See [Password Group \$n\$ - Lock 3 Status Register \(PASS_LOCK3_PG \$n\$ \)](#)). Both these debug enabling options are further conditioned by life cycle and Production Disable.

The debug interface enable truth table for this chip is shown in the device specific PASS section of this chapter.

67.5.2.1 JTAG Password Challenge

To enable the Debug Interface Access, the JTAG password challenge register in the JTAGC, needs to be written by an external tool. If that password matches the JTAG password stored in UTEST flash, Debug Interface Access is enabled until the next destructive reset.

67.5.3 Flash Memory Read Protection

Flash Memory Read Protection is a security feature that can selectively make regions of the flash memory unreadable. Any attempt to read a region of flash that is read protected results in the read access terminating with a transfer error. This is consistent for all bus masters including debug bus masters.

The assertion of flash memory region read protection is controlled by Life Cycle, Censorship and whether the Debug Interface access is enabled or blocked. The Flash memory read protection truth table describes the combination of conditions that drive Flash Memory Read Protection.

The intention of the feature is to protect the contents of the flash from being read. The flash memory is readable in normal operation, allowing the application to run, but becomes read protected when a Debug tool is attached and enabled (JTAG password or LOCK3[DBL] bit).

The flash memory is also read protected when the life cycle status is Failure Analysis. This life cycle is used for a module field failure, where the device is returned for failure analysis. This protection prevents the contents of the flash memory from being read.

The flash memory is divided into a number of Regions (3 to 5 depending on the device). The assignment of the regions is defined in the device reference manual.

The memory read protection truth table for this chip is shown in the device specific PASS information.

67.5.4 Production Disable

67.5.4.1 Overview and rationale

Production Disable is an optional feature that, once enabled, allows the MCU to function completely normally without any constraints, except the Debug interface is disabled.

67.5.4.2 Enabling

The feature is enabled via the Production Disable DCF client in the PASS module. Bit 0 enables the Debug Interface Disable. See [Production Disable](#) for details.

This information can be written by adding a DCF Record to the DCF record list in Flash UTEST. The DCF Client is configured to accept only one write. Subsequent DCF records that write to this DCF client are ignored.

If no DCF Record writes to this client, the default state of this feature is disabled. If, for a particular application, there is no intention of ever using this feature, it would be good practice to add a DCF Record that writes this client to 0x0000_0000. This is not essential, but ensures the feature can not be enabled accidentally.

67.6 Initialization/application information

67.6.1 Reset

The reset state of each individual bit is shown within the Register Description section. However, many bits receive their values based on DCF records stored in the UTEST Flash memory block.

67.6.2 Setting lock bits in a password group

There is no restriction on setting and clearing lock registers in the UTEST Flash, apart from the available space for DCF records. It is possible to set and clear them multiple times.

Table 67-38. CS/ADDR Values for Lock Registers

Register	DCF Command Word
PW Group 0	
Lock0	0xxxxx ¹ _0100
Lock1	0xxxxx ¹ _0104
Lock2	0xxxxx ¹ _0108
Lock3	0xxxxx ¹ _010C
PW Group 1	
Lock0	0xxxxx ¹ _0110

Table continues on the next page...

Table 67-38. CS/ADDR Values for Lock Registers (continued)

Register	DCF Command Word
Lock1	0xxxxx ¹ _0114
Lock2	0xxxxx ¹ _0118
Lock3	0xxxxx ¹ _011C
PW Group 2	
Lock0	0xxxxx ¹ _0120
Lock1	0xxxxx ¹ _0124
Lock2	0xxxxx ¹ _0128
Lock3	0xxxxx ¹ _012C
PW Group3	
Lock0	0xxxxx ¹ _0130
Lock1	0xxxxx ¹ _0134
Lock2	0xxxxx ¹ _0138
Lock3	0xxxxx ¹ _013C

1. Value depends on the DCF Client Select assigned to the PASS module in the device. See chip specific PASS information for the Client Select assignment

In order to set the MIDLOCK field in the Lock0 register of password group 1 to the value 0x00FF and other fields to 0, the DCF record has to be added as shown in [Table 67-39](#).

Table 67-39. Setting some bits in the MIDLOCK field

ADDR offset	DATA	Description
0x00	0x05AA_55AF	Start Record
0x04	0x0000_0000	
0x08	0x0000_00FF	PW Group 1, Lock0 Set MIDLOCK to 0xFF
0x0C	0xxxxx_0110	
0x10	0xFFFF_FFFF	Non-Programmed Addresses
0x14	0xFFFF_FFFF	
...	...	

Chapter 68

Tamper Detection Module (TDM)

68.1 Chip-specific TDM information

This chip has one instance of the TDM. The TDM and its associated DCF clients enable the user to:

- Select a flash memory block for tamper detection protection by assigning the block to a Tamper Detection Region (TDR)
- Override a block's tamper detection protection under certain circumstances
- Disable the override of a block's tamper detection protection
- Configure a flash memory block as One Time Programmable (OTP), meaning the block can be written only once

CAUTION

Specifying flash memory blocks to be monitored by the TDM and configuring flash memory blocks as OTP is done by defining DCF records. Use caution when assigning flash memory blocks to TDRs or configuring blocks as OTP. Some DCF records can be written only one time. Many DCF records can be written more than once, but the setting defined in each write is irreversible. For details about TDM-related DCF records and DCF clients, see [DCF clients](#). For additional information, see the chip-level [Device Configuration Format \(DCF\) Records](#) details.

68.1.1 Disabling the override of tamper detection

The [Tamper Region Override \(TO\) DCF client](#) allows the user to override tamper detection protection. The override cancels tamper detection protection for all blocks in a TDR and is irreversible.

The [Software Tamper Region Override Disable \(STO_DIS\) DCF client](#) allows the user to disable the override option permanently.

To optimize security on this chip, disable the Tamper Region Override option permanently by using the Software Tamper Region Override Disable DCF client.

68.1.2 TDR_n_LOCK_m DCF client bit mapping

The TDR_n_LOCK_m DCF clients provide a means to select flash memory blocks for inclusion in a TDR. Specific memory blocks map to particular bits in TDR_n_LOCK_m clients.

68.1.2.1 TDR_n_LOCK0 DCF client bit mapping

The flash memory blocks mapped to the [Tamper Region Assignment DCF client \(TDR_n_LOCK0\)](#) bits are as follows.

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
Register Field	Reserved	Reserved	Reserved	Reserved	LOWTDRAGN[11:0]											MIDTDRAGN[15:0]																
Flash Memory Block Name	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—
64 KB EEPROM Data Block1 - Partition 1																																
64 KB EEPROM Data Block0 - Partition 0																																
64 KB EEPROM Data Block1 - Partition 1																																
64 KB EEPROM Data Block0 - Partition 2																																

68.1.2.2 TDRn_LOCK1 DCF client bit mapping

The flash memory blocks mapped to the [Tamper Region Assignment DCF client \(TDRn_LOCK1\)](#) bits are as follows.

NOTE

Because the CSE module has exclusive access to these flash memory blocks, this DCF client and the associated DCF records are ignored.

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	
Register Field	Reserved															HIGHTDRAGN[15:0]																	
Flash Memory Block Name	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	16 KB CSE Data Block1 - Partition 5	16 KB CSE Data Block0 - Partition 4

68.1.2.3 TDRn_LOCK2 DCF client bit mapping

The flash memory blocks mapped to the [Tamper Region Assignment DCF client \(TDRn_LOCK2\)](#) bits are as follows.

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
Register Field	256KBTDRAIN[31:0]																															

Table continues on the next page...

Flash Memory Block Name
256 KB Application Code Block31 - Partition 9
256 KB Application Code Block30 - Partition 9
256 KB Application Code Block29 - Partition 9
256 KB Application Code Block28 - Partition 9
256 KB Application Code Block27 - Partition 9
256 KB Application Code Block26 - Partition 9
256 KB Application Code Block25 - Partition 9
256 KB Application Code Block24 - Partition 9
256 KB Application Code Block23 - Partition 8
256 KB Application Code Block22 - Partition 8
256 KB Application Code Block21 - Partition 8
256 KB Application Code Block20 - Partition 8
256 KB Application Code Block19 - Partition 8
256 KB Application Code Block18 - Partition 8
256 KB Application Code Block17 - Partition 8
256 KB Application Code Block16 - Partition 8
256 KB Application Code Block15 - Partition 7
256 KB Application Code Block14 - Partition 7
256 KB Application Code Block13 - Partition 7
256 KB Application Code Block12 - Partition 7
256 KB Application Code Block11 - Partition 7
256 KB Application Code Block10 - Partition 7
256 KB Application Code Block9 - Partition 7
256 KB Application Code Block8 - Partition 7
256 KB Application Code Block7 - Partition 6
256 KB Application Code Block6 - Partition 6
256 KB Application Code Block5 - Partition 6
Calibration Block4 - Partition 6
Calibration Block3 - Partition 6
Calibration Block2 - Partition 6
Calibration Block1 - Partition 6
Boot Code Block0 - Partition 6

68.1.2.4 TDR_n_LOCK3 DCF client bit mapping

The flash memory blocks mapped to the [Tamper Region Assignment DCF client \(TDR_n_LOCK3\)](#) bits are as follows.

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
Register Field	Reserved															256KBTDRAGN[15:0]																
Flash Memory Block Name																																

68.1.3 OTPEN_n DCF client bit mapping

The OTPEN_n DCF clients provide a means to configure flash memory blocks to be One Time Programmable (OTP). Specific memory blocks map to particular bits in OTPEN_n clients.

68.1.3.1 OTPEN0 DCF client bit mapping

The flash memory blocks mapped to the [One Time Programmable Enable \(OTPEN0\) DCF client](#) bits are as follows.

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
Register Field	Reserved	Reserved	Reserved	Reserved	LOWOTPEN[11:0]											MIDOTPEN[15:0]																
Flash Memory Block Name	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—
64 KB EEPROM Data Block1 - Partition 1																																
64 KB EEPROM Data Block0 - Partition 0																																
—																																
—																																
—																																
—																																
—																																
—																																
—																																
—																																
—																																
—																																
64 KB EEPROM Data Block1 - Partition 3																																
64 KB EEPROM Data Block0 - Partition 2																																

68.1.3.2 OTPEN1 DCF client bit mapping

The flash memory blocks mapped to the [One Time Programmable Enable \(OTPEN1\) DCF client](#) bits are as follows.

NOTE

When the value of a bit in this DCF client reflects that the corresponding flash block is one-time programmable (OTP),

Chip-specific TDM information

the OTP status applies to the block even though the CSE module has exclusive access to the block.

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	
Register Field	Reserved															HIGHOTPEN[15:0]																	
Flash Memory Block Name	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	16 KB CSE Data Block1 - Partition 5	16 KB CSE Data Block0 - Partition 4

68.1.3.3 OTPEN2 DCF client bit mapping

The flash memory blocks mapped to the [One Time Programmable Enable \(OTPEN2\) DCF client](#) bits are as follows.

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
Register Field	256KBOTPEN[31:0]																															

Table continues on the next page...

Flash Memory Block Name
256 KB Application Code Block31 - Partition 9
256 KB Application Code Block30 - Partition 9
256 KB Application Code Block29 - Partition 9
256 KB Application Code Block28 - Partition 9
256 KB Application Code Block27 - Partition 9
256 KB Application Code Block26 - Partition 9
256 KB Application Code Block25 - Partition 9
256 KB Application Code Block24 - Partition 9
256 KB Application Code Block23 - Partition 8
256 KB Application Code Block22 - Partition 8
256 KB Application Code Block21 - Partition 8
256 KB Application Code Block20 - Partition 8
256 KB Application Code Block19 - Partition 8
256 KB Application Code Block18 - Partition 8
256 KB Application Code Block17 - Partition 8
256 KB Application Code Block16 - Partition 8
256 KB Application Code Block15 - Partition 7
256 KB Application Code Block14 - Partition 7
256 KB Application Code Block13 - Partition 7
256 KB Application Code Block12 - Partition 7
256 KB Application Code Block11 - Partition 7
256 KB Application Code Block10 - Partition 7
256 KB Application Code Block9 - Partition 7
256 KB Application Code Block8 - Partition 7
256 KB Application Code Block7 - Partition 6
256 KB Application Code Block6 - Partition 6
256 KB Application Code Block5 - Partition 6
Calibration Block4 - Partition 6
Calibration Block3 - Partition 6
Calibration Block2 - Partition 6
Calibration Block1 - Partition 6
Boot Code Block0 - Partition 6

68.1.3.4 OTPEN3 DCF client bit mapping

The flash memory blocks mapped to the [One Time Programmable Enable \(OTPEN3\) DCF client](#) bits are as follows.

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
Register Field	Reserved															256KBOTPEN[15:0]																
Flash Memory Block Name																																

68.2 Overview

The Tamper Detection Module provides a type of flash memory erase protection mechanism that requires software to write a record associated with one or more blocks in a Tamper Detection Region (TDR) before the block(s) can be erased. The mechanism does not *prohibit* an erase operation on any block within a TDR—it requires a record to be written before the operation can execute. Collectively, the records are referred to as a “diary”. At minimum, the diary serves as an erase counter. Because the diary record content is customer-defined, the diary can also serve as a history.

Up to 6 TDRs can be defined via DCF records.

The following figure shows the block diagram of the TDM and its interface to other system components.

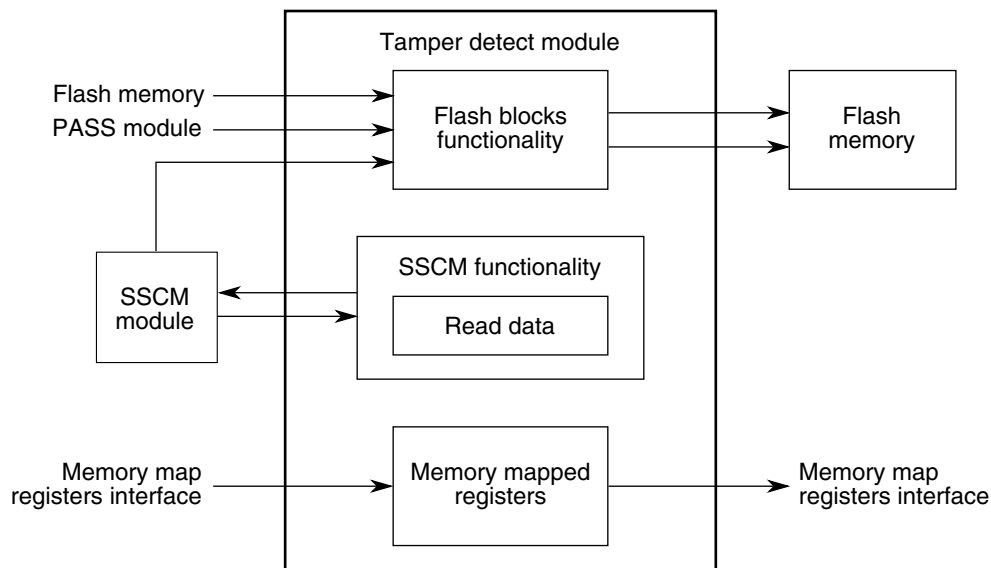


Figure 68-1. Tamper Detection Module block diagram

Security information is passed to the TDM by the SSCM module via the DCF bus. Among the data are:

- OTP (One Time Programmable) settings for each of the flash blocks
- Base address for the diary in the flash
- Block assignments for the TDRs
- Tamper region override information
- Software tamper region disable

The TDM module controls the flash blocks for erase operations and sets them as OTP.

The two flash buses are used to control the flash blocks for erase operation and set them as OTP. OTP means that flash erase of the entire block is disabled and the only words that are already erased can be programmed. Over-programming is not possible.

NOTE

The pass module provides the control for erase operation for each flash block as well as for each region of flash memory, and the PASS settings have priority over the settings of TDM.

The flash block indicates to the TDM the address of the programmed word. This address is internally decoded and provides a means to verify whether the block is part of any TDRs. The memory map registers interface provides a means to the user to read information of the TDM registers.

68.3 Features

The Tamper Detection Module supports these distinctive features:

- Erase counter (diary) for each TDR to record up to 256 write attempts on the flash blocks
 - The diary is divided into 6 parts (TDRs), each consisting of 256 64-bit records.
 - The diary block should be assigned as OTP region in flash.
 - The diary area can store any type of relevant information.
- Tamper Detection Region Block Assignment
 - Any flash block can be assigned to any TDR and be controlled.
- 32-bit Erase Lock DCF clients for each flash memory TDR defined.
- 32-bit OTP Enable DCF clients to disable the erase operation for an entire flash block
- Override TDR via DCF records¹
 - Allows block assigned to a TDR to be unlocked for erase even if the TDR is locked for this operation.
- Memory mapped registers interface for reading via software of specific registers.

68.4 External signal description

There are no external signals driving or being driven by the TDM.

1. The Tamper Region Override DCF record is a "write one only" DCF record. If an override is applied to a TDR it is permanent and cannot be reversed.

68.5 DCF clients

TDM-related DCF records perform the following functions:

- Establish a permanent diary base address
- Define Tamper Detect Regions (TDRs) to monitor on-chip flash memory program/erase activity
- Permanently disable one or more Tamper Regions
- Permanently disable ability for software to permanently override one or more TDRs
- Configure flash memory as One Time Programmable (OTP) on a per-block basis

The following table gives an overview on the TDM DCF clients implemented.

Table 68-1. DCF clients

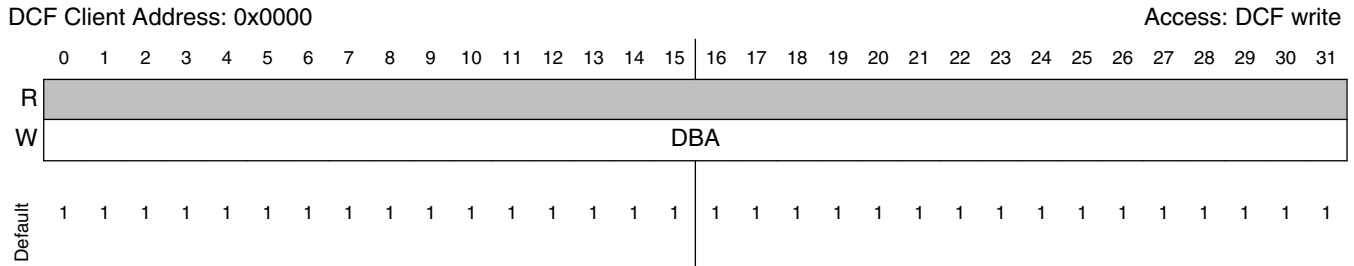
DCF Client Name	Description	Strategy
Diary Base Address	Stores the base address of the tamper detection diary	write once, write 1 to 0 only
Tamper Region Override	Stores information of which TDR is to be overridden	write 0 to 1 only
Software Tamper Region Override Disable	Stores information to disable the software mechanism to override the TDR	write once, write 0 to 1 only
OTPEN[nb] (OTP Enable)	Stores information of which flash blocks is to be assigned as OTP (One Time Programmable) - nb means one register for each Lock register of the flash memory	write 0 to 1 only
TDR[n]_LOCK[m] Region Assignment	Stores information of which flash block is assigned to a TDR. • n specifies a TDR • m specifies the flash blocks as per the LOCK registers of the flash memory.	write 0 to 1 only

68.5.1 Diary Base Address (DBA) DCF client

This DCF client holds base address information of the diary. The following diagram depicts this client.

CAUTION

This write-once DCF client can be written only one time. After one DCF record has written to this client, all subsequent writes are ignored.

**Table 68-2. Diary Base Address (DBA) DCF client field description**

Field	Description
0–31 DBA	Diary Base Address. This field holds the base address of the diary. This information is then used to verify if the diary being programmed matches any TDR region. NOTE: Diary Base Address must be on a 16 KB boundary.

68.5.2 Tamper Region Override (TO) DCF client

This DCF client holds override information for each TDR. The following figure shows the diagram for this register.

CAUTION

This DCF client is writable from 0 to 1 only. Attempted writes of bits from 1 to 0 are ignored, making the override of a TDR permanent.

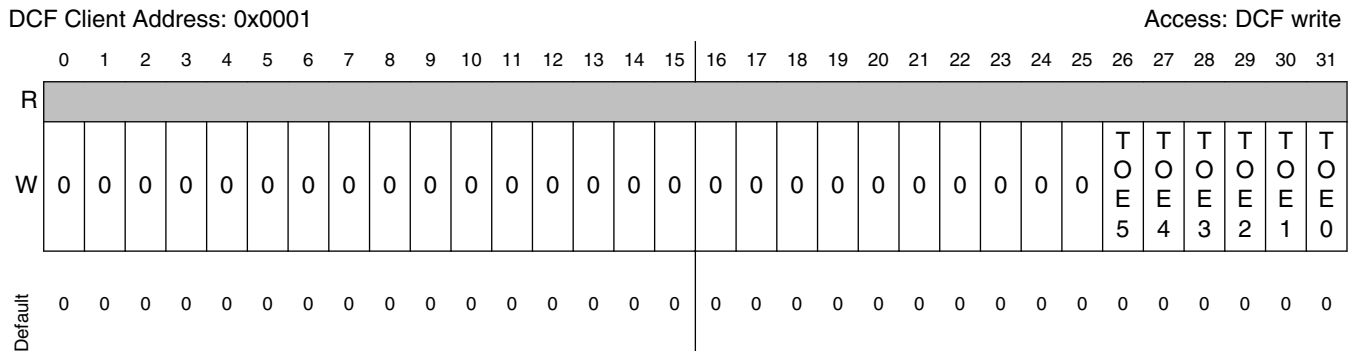


Table 68-3. Tamper Region Override (TO) DCF client field descriptions

Field	Description
0–25 Reserved	This field is reserved and should be written with all zeroes when programming the TOE fields.
26–31 TOE5 to TOE0	TDR Override Enable. Each TOE _n field, where <i>n</i> represents the TDRs, holds information of the Override Enable status for each TDR.
0	Normal operation. Blocks assigned to this TDR can NOT be erased until the diary is written.
1	Diary override. The blocks assigned to this TDR can be erased WITHOUT writing to the diary.

68.5.3 Software Tamper Region Override Disable (STO_DIS) DCF client

This DCF client holds information to disable the software mechanism to override the tamper detect regions.

CAUTION

This write-once DCF client can be written only one time. After one DCF record has written to this client, all subsequent writes are ignored.

DCF Client Address: 0x0002

Access: DCF write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R																																
W	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Default	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 68-4. Software Tamper Region Override Disable (STO_DIS) DCF client field descriptions

Field	Description
0–25 Reserved	This field is reserved and should be written with all zeroes when programming the STO_DIS fields.

Table continues on the next page...

Table 68-4. Software Tamper Region Override Disable (STO_DIS) DCF client field descriptions (continued)

Field	Description
26–31 STO_DIS5 to STO_DIS0	Software TDR Override Disable TDR _n . Each STO_DIS _n field, where <i>n</i> represents the TDR number, holds information to disable the software mechanism to override TDR _n .
0	Software tamper detect override mechanism supported
1	Software tamper detect override mechanism disabled. Any attempt to override TDR _n by loading a service key value in STO_KEY _n register is ignored.

68.5.4 One Time Programmable Enable (OTPEN0) DCF client

This DCF client holds OTP information for particular blocks of flash memory. This DCF client mirrors the LOCK0 register of the flash memory. The following diagram depicts this client.

CAUTION

This DCF client is writable from 0 to 1 only. Attempted writes of bits from 1 to 0 are ignored, making the OTP assignment of a flash memory block irreversible.

DCF Client Address: 0x0008

Access: DCF write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R																																
W	0	0	0	0	LOWOTPEN[11:0]											MIDOTPEN[15:0]																
Default	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	

Table 68-5. One Time Programmable Enable (OTPEN0) DCF client

Field	Description
0–3 Reserved	This field is reserved and should be written with all zeroes when writing to this client.
4–15 LOWOTPEN	Low Block OTP Enable. This field affects flash memory blocks in Low address space. See the chip-specific information for the mapping between field bits and flash blocks. The same mapping also applies to control similar functions in the flash memory module.
0	Corresponding flash blocks are not assigned as OTP
1	Corresponding flash blocks are assigned as OTP
16–31 MIDOTPEN	Mid Block OTP Enable. This field affects flash memory blocks in Mid address space. See the chip-specific information for the mapping between field bits and flash blocks. The same mapping also applies to control similar functions in the flash memory module.
0	Corresponding flash blocks are not assigned as OTP
1	Corresponding flash blocks are assigned as OTP

68.5.5 One Time Programmable Enable (OTPEN1) DCF client

This DCF client holds OTP information for particular blocks of flash memory. This DCF client mirrors the LOCK1 register of the flash memory. The following diagram depicts the client.

CAUTION

This DCF client is writable from 0 to 1 only. Attempted writes of bits from 1 to 0 are ignored, making the OTP assignment of a flash memory block irreversible.

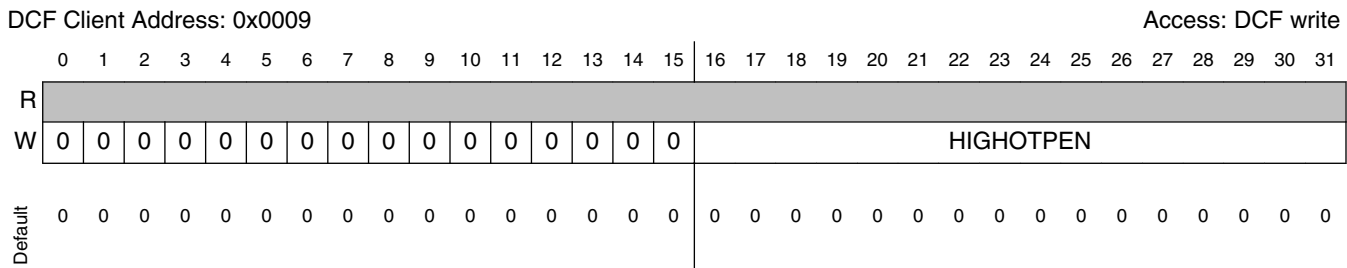


Table 68-6. One Time Programmable Enable (OTPEN1) DCF client field descriptions

Field	Description
0–15 Reserved	This field is reserved and should be written with zeroes when writing to this client.
16–31 HIGHOTPEN	High Block OTP Enable. This field affects flash memory blocks in High address space. See the chip-specific information for the mapping between field bits and flash blocks. The same mapping also applies to control similar functions in the flash memory module. 0 Corresponding flash blocks are not assigned as OTP 1 Corresponding flash blocks are assigned as OTP

68.5.6 One Time Programmable Enable (OTPEN2) DCF client

This DCF client holds OTP information for particular blocks of flash memory. This DCF client mirrors the LOCK2 register of the flash memory. The following diagram depicts the client.

CAUTION

This DCF client is writable from 0 to 1 only. Attempted writes of bits from 1 to 0 are ignored, making the OTP assignment of a flash memory block irreversible.

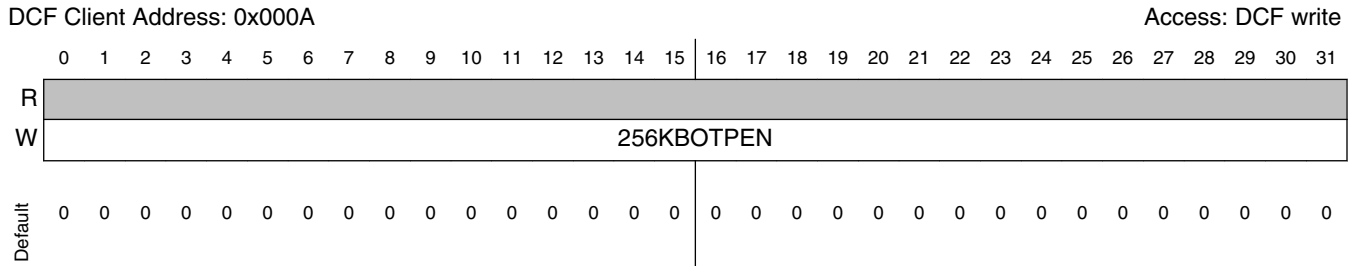


Table 68-7. One Time Programmable Enable (OTPEN2) DCF client field descriptions

Field	Description
0–31 256KBOTPEN	256 KB Block OTP Enable. This field affects flash memory blocks in 256KB address space. See the chip-specific information for the mapping between field bits and flash blocks. The same mapping also applies to control similar functions in the flash memory module.
0	Corresponding flash blocks are not assigned as OTP
1	Corresponding flash blocks are assigned as OTP

68.5.7 One Time Programmable Enable (OTPEN3) DCF client

This DCF client holds OTP information for particular blocks of flash memory. This DCF client mirrors the LOCK3 register of the flash memory. The following diagram depicts the client.

CAUTION

This DCF client is writable from 0 to 1 only. Attempted writes of bits from 1 to 0 are ignored, making the OTP assignment of a flash memory block irreversible.

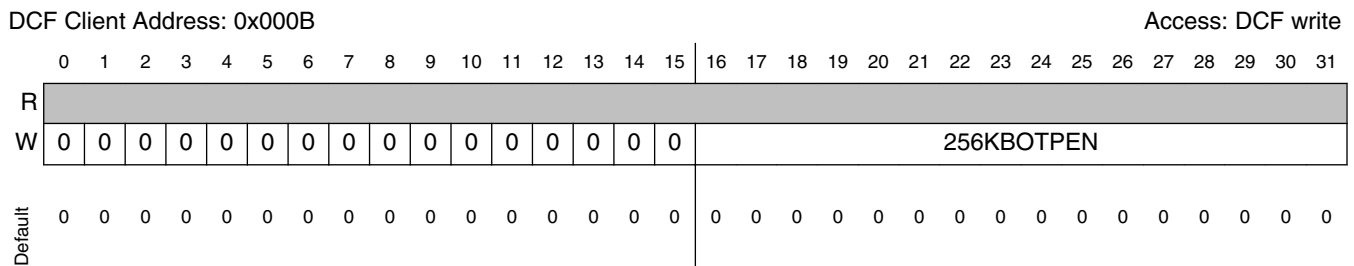


Table 68-8. One Time Programmable Enable (OTPEN3) DCF client field descriptions

Field	Description
0–15 Reserved	This field is reserved and should be written with zeroes when writing to this client.
16–31 256KBOTPEN	256 KB Block OTP Enable. This field affects flash memory blocks in the upper range of 256KB address space. See the chip-specific information for the mapping between field bits and flash blocks. The same mapping also applies to control similar functions in the flash memory module.
0	Corresponding flash blocks are not assigned as OTP
1	Corresponding flash blocks are assigned as OTP

68.5.8 Tamper Region Assignment DCF client (TDRn_LOCK0)

This DCF client holds the assignment information of flash memory blocks to a specific TDR. Each flash memory block can be assigned to any TDR (n varies from 0 to 5). This DCF client holds the assignment information for only those blocks specified in the LOCK0 register in the flash memory. For that reason, the layout of this DCF client mirrors the LOCK0 register in the flash memory. The following diagram depicts the TDRn_LOCK0 DCF client.

CAUTION

This DCF client is writable from 0 to 1 only. Attempted writes of bits from 1 to 0 are ignored, making the assignment of a flash memory block to a TDR irreversible unless the entire TDR is overridden.

DCF Client Address: $0x0014 + ((n \times 4) + 0)$

Access: DCF write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	
R																																	
W	0	0	0	0	LOWTDRAGN												MIDTDRAGN																
Default	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0		

Table 68-9. Tamper Region Assignment DCF client (TDRn_LOCK0)

Field	Description
0–3 Reserved	This field is reserved and should be written with zeroes when writing to this client.

Table continues on the next page...

Table 68-9. Tamper Region Assignment DCF client (TDRn_LOCK0) (continued)

Field	Description
4–15 LOWTDRAGN	Low Block Tamper Region Assignment. This field affects flash memory blocks in Low address space. See the chip-specific information for the mapping between field bits and flash blocks. The same mapping also applies to control similar functions in the flash memory module.
	0 Corresponding flash blocks are not assigned to the TDRn
	1 Corresponding flash blocks are assigned to the TDRn
16–31 MIDTDRAGN	Mid Block Tamper Region Assignment. This field affects flash memory blocks in Mid address space. See the chip-specific information for the mapping between field bits and flash blocks. The same mapping also applies to control similar functions in the flash memory module.
	0 Corresponding flash blocks are not assigned to the TDRn
	1 Corresponding flash blocks are assigned to the TDRn

68.5.9 Tamper Region Assignment DCF client (TDRn_LOCK1)

This DCF client holds the assignment information of flash memory blocks to a specific TDR. Each flash memory block can be assigned to any TDR (n varies from 0 to 5). This DCF client holds the assignment information for only those blocks specified in the LOCK1 register in the flash memory. For that reason, the layout of this DCF client mirrors the LOCK1 register in the flash memory. The following diagram depicts the TDRn_LOCK1 DCF client.

CAUTION

This DCF client is writable from 0 to 1 only. Attempted writes of bits from 1 to 0 are ignored, making the assignment of a flash memory block to a TDR irreversible unless the entire TDR is overridden.

DCF Client Address: $0x0014 + ((n \times 4) + 1)$

Access: DCF write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R																																
W	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	HIGHTDRAGN																
Default	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	

Table 68-10. Tamper Region Assignment DCF client (TDRn_LOCK1)

Field	Description
0–15 Reserved	This field is reserved and should be written with zeroes when writing to this client.

Table continues on the next page...

Table 68-10. Tamper Region Assignment DCF client (TDRn_LOCK1) (continued)

Field	Description
16–31 HIGHTDRAGN	High Block Tamper Region Assignment. This field affects flash memory blocks in High address space. See the chip-specific information for the mapping between field bits and flash blocks. The same mapping also applies to control similar functions in the flash memory module.
0	Corresponding flash blocks are not assigned to the TDRn
1	Corresponding flash blocks are assigned to the TDRn

68.5.10 Tamper Region Assignment DCF client (TDRn_LOCK2)

This DCF client holds the assignment information of flash memory blocks to a specific TDR. Each flash memory block can be assigned to any TDR (n varies from 0 to 5). This DCF client holds the assignment information for only those blocks specified in the LOCK2 register in the flash memory. For that reason, the layout of this DCF client mirrors the LOCK2 register in the flash memory. The following diagram depicts the TDRn_LOCK2 DCF client.

CAUTION

This DCF client is writable from 0 to 1 only. Attempted writes of bits from 1 to 0 are ignored, making the assignment of a flash memory block to a TDR irreversible unless the entire TDR is overridden.

DCF Client Address: $0x0014 + ((n \times 4) + 2)$

Access: DCF write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R																																
W	256KBTDRAGN[31:0]																															
Default	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 68-11. Tamper Region Assignment DCF client (TDRn_LOCK2)

Field	Description
0–31 256KBTDRAGN[31:0]	256 KB Block Tamper Region Assignment. This field affects flash memory blocks in 256KB address space. See the chip-specific information for the mapping between field bits and flash blocks. The same mapping also applies to control similar functions in the flash memory module.
0	Corresponding flash blocks are not assigned to the TDRn
1	Corresponding flash blocks are assigned to the TDRn

68.5.11 Tamper Region Assignment DCF client (TDRn_LOCK3)

This DCF client holds the assignment information of flash memory blocks to a specific TDR. Each flash memory block can be assigned to any TDR (n varies from 0 to 5). This DCF client holds the assignment information for only those blocks specified in the LOCK3 register in the flash memory. For that reason, the layout of this DCF client mirrors the LOCK3 register in the flash memory. The following diagram depicts the TDRn_LOCK3 DCF client.

CAUTION

This DCF client is writable from 0 to 1 only. Attempted writes of bits from 1 to 0 are ignored, making the assignment of a flash memory block to a TDR irreversible unless the entire TDR is overridden.

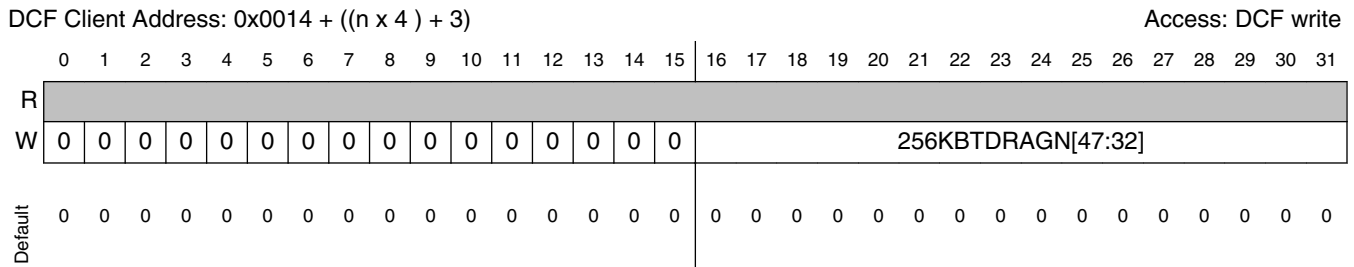


Table 68-12. Tamper Region Assignment DCF client (TDRn_LOCK3)

Field	Description
0–15 Reserved	This field is reserved and should be written with zeroes when writing to this client.
16–31 256KBTDRAgn[47:32]	256KB Block Tamper Region Assignment. This field affects flash memory blocks in 256KB address space. See the chip-specific information for the mapping between field bits and flash blocks. The same mapping also applies to control similar functions in the flash memory module.
0	Corresponding flash blocks are not assigned to the TDRn
1	Corresponding flash blocks are assigned to the TDRn

68.6 Memory Map/Register Definition

TDM memory map

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
0	TDR Status Register (TDM_TDRSR)	32	R	See section	68.6.1/3340
4	Last Flash Programmed Address Register (TDM_LFPAR)	32	R	0000_0000h	68.6.2/3342
8	Diary Base Address (TDM_DBA)	32	R	See section	68.6.3/3343
10	Software Tamper Override Key Region (TDM_STO_KEY0)	32	R/W	0000_0000h	68.6.4/3343
14	Software Tamper Override Key Region (TDM_STO_KEY1)	32	R/W	0000_0000h	68.6.4/3343
18	Software Tamper Override Key Region (TDM_STO_KEY2)	32	R/W	0000_0000h	68.6.4/3343
1C	Software Tamper Override Key Region (TDM_STO_KEY3)	32	R/W	0000_0000h	68.6.4/3343
20	Software Tamper Override Key Region (TDM_STO_KEY4)	32	R/W	0000_0000h	68.6.4/3343
24	Software Tamper Override Key Region (TDM_STO_KEY5)	32	R/W	0000_0000h	68.6.4/3343

68.6.1 TDR Status Register (TDM_TDRSR)

This register contains the current status, i.e., locked or unlocked, of each tamper region, i.e., TDR.

- If a tamper region is locked, the blocks assigned to that tamper region cannot be erased.
- If a region is unlocked, the blocks within the tamper region can be erased.

The register contains a status bit for each TDR. Status is determined as follows:

1. The default status for each TDR is '1', which indicates that the erase operation is disabled for all flash blocks associated with the TDR.
2. The status bit for a TDR is set to '0' (indicating that the erase operation for all flash blocks associated with the TDR is enabled) if either a successful programming operation to the TDR diary region has been performed or the TDR Override bit for that TDR is set.

NOTE

- TDRSR is a read-only register and writes have no effect on it. Write operations result in a transfer error.
- If the reset value for this register is determined by a DCF record, it will be noted in the chip-specific information at the beginning of this chapter. Otherwise, the reset value is as shown.

Address: 0h base + 0h offset = 0h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0										TDRSR5	TDRSR4	TDRSR3	TDRSR2	TDRSR1	TDRSR0
W																
Reset	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1

TDM_TDRSR field descriptions

Field	Description
0–25 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
26 TDRSR5	Represents the status of TDR 5 in conjunction with the Diary Override Enable status. 0 TDR is unlocked. Blocks assigned to this TDR are NOT blocked for erasing by the TDM module. This unlock state is achieved by a successful programming operation to the correct diary region or this TDR has been permanently unlocked by a DCF record that writes to the Tamper Region Override DCF register or a successful software override operation. 1 TDR is locked. Erase of blocks assigned to this TDR is not possible.
27 TDRSR4	Represents the status of TDR 4 in conjunction with the Diary Override Enable status. 0 TDR is unlocked. Blocks assigned to this TDR are NOT blocked for erasing by the TDM module. This unlock state is achieved by a successful programming operation to the correct diary region or this TDR has been permanently unlocked by a DCF record that writes to the Tamper Region Override DCF register or a successful software override operation. 1 TDR is locked. Erase of blocks assigned to this TDR is not possible.
28 TDRSR3	Represents the status of TDR 3 in conjunction with the Diary Override Enable status. 0 TDR is unlocked. Blocks assigned to this TDR are NOT blocked for erasing by the TDM module. This unlock state is achieved by a successful programming operation to the correct diary region or this TDR

Table continues on the next page...

TDM_TDRSR field descriptions (continued)

Field	Description
	has been permanently unlocked by a DCF record that writes to the Tamper Region Override DCF register or a successful software override operation. 1 TDR is locked. Erase of blocks assigned to this TDR is not possible.
29 TDRSR2	Represents the status of TDR 2 in conjunction with the Diary Override Enable status. 0 TDR is unlocked. Blocks assigned to this TDR are NOT blocked for erasing by the TDM module. This unlock state is achieved by a successful programming operation to the correct diary region or this TDR has been permanently unlocked by a DCF record that writes to the Tamper Region Override DCF register or a successful software override operation. 1 TDR is locked. Erase of blocks assigned to this TDR is not possible.
30 TDRSR1	Represents the status of TDR 1 in conjunction with the Diary Override Enable status. 0 TDR is unlocked. Blocks assigned to this TDR are NOT blocked for erasing by the TDM module. This unlock state is achieved by a successful programming operation to the correct diary region or this TDR has been permanently unlocked by a DCF record that writes to the Tamper Region Override DCF register or a successful software override operation. 1 TDR is locked. Erase of blocks assigned to this TDR is not possible.
31 TDRSR0	Represents the status of TDR 0 in conjunction with the Diary Override Enable status. 0 TDR is unlocked. Blocks assigned to this TDR are NOT blocked for erasing by the TDM module. This unlock state is achieved by a successful programming operation to the correct diary region or this TDR has been permanently unlocked by a DCF record that writes to the Tamper Region Override DCF register or a successful software override operation. 1 TDR is locked. Erase of blocks assigned to this TDR is not possible.

68.6.2 Last Flash Programmed Address Register (TDM_LFPAR)

This register holds information of the address of the last successful programming operation provided from the flash memory to the TDM. Writes to this register are ignored and will generate a transfer error.

Address: 0h base + 4h offset = 4h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	LFPAR																															
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	

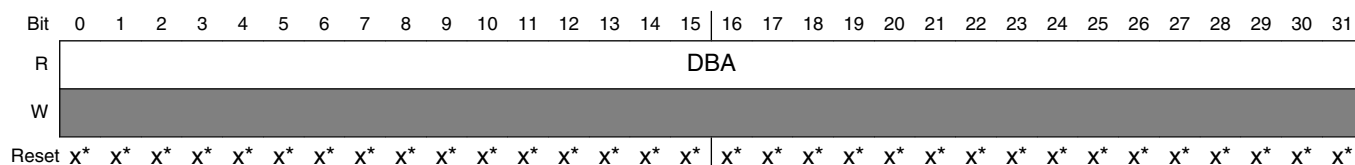
TDM_LFPAR field descriptions

Field	Description
0–31 LFPAR	Last Flash Programmed Address. These bits represent the address of the last successful programming operation.

68.6.3 Diary Base Address (TDM_DBA)

This register holds base address information of the diary. Any write to this register is ignored and generates a transfer error.

Address: 0h base + 8h offset = 8h



* Notes:

- The reset value of this register is set via DCF record.x = Undefined at reset.

TDM_DBA field descriptions

Field	Description
0–31 DBA	Diary Base Address. These bits represent the address of the diary location in the flash. NOTE: The diary base address is established via the DBA DCF record. The reset value of this register is determined by that record.

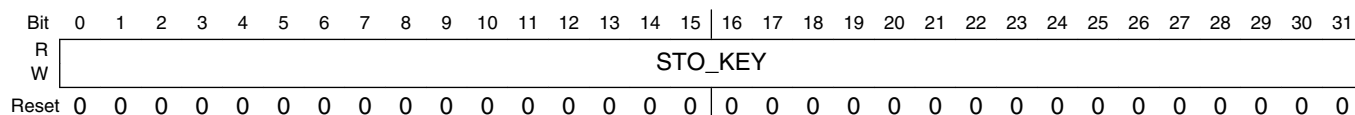
68.6.4 Software Tamper Override Key Region (TDM_STO_KEYn)

The STO_KEYn register holds the service key for overriding the Tamper Detect Region n, where n denotes the Tamper Detect Region number.

Caution

- The STO_KEYn registers can only be written once.
- The associated DCF client is writable from 0 to 1 only.
Attempted writes of bits from 1 to 0 are ignored.

Address: 0h base + 10h offset + (4d × i), where i=0d to 5d



TDM_STO_KEYn field descriptions

Field	Description
0–31 STO_KEY	Software Tamper Override Key Diary n. This field holds the service key value for overriding Tamper Detect Region n. By loading STO_KEYn with the correct signature of 0x55AA5A5A, Tamper Detect Region n is overridden, and the flash blocks assigned to Tamper Detect Region n are open for erase. This field has no effect when STO_DISn field of STO_DIS DCF client is set. Caution: The STO_KEYn registers can only be written once.

TDM_STO_KEY n field descriptions (continued)

Field	Description

68.7 Functional description

The primary purpose of the TDM is to provide a Tamper Detection mechanism by enabling an Erase Counter (or diary) for regions within the flash. These regions are called Tamper Detection Regions. The following sections cover the entire functionality of this module.

68.7.1 Flash erase counter and tamper detection

A flash region block or part of it is assigned for implementing the diary. The base address for the diary is defined by a DCF Record.

The diary is divided into 6 parts where each part consists of 256×64 -bit records, to record up to 256 write attempts on the flash blocks assigned to each Tamper Detection Region. For correct operation, the diary should reside in a flash block that is assigned as OTP.

Each flash block can be assigned to any of the TDRs. Related to each TDR is a group of 4×32 -bit internal hardware registers that are initialized by DCF records that define the tamper region. These registers mirror exactly the LOCK registers bits from the flash module.

Each TDR n _LOCK m register is loaded at boot time by the SSCM, which defines if the associated flash block is Locked for Erase in a particular TDR:

- '1' means the flash block is locked for erase
- '0' means the flash block is not locked for erase

Before performing an erase in one of the blocks locked for erase and associated to a TDR, a program operation has to be performed on the diary part for that TDR. As the diary should be OTP, successive writes are allowed on unprogrammed double words only. This ensures that diary records can only increase and that data recorded in the diary is not modifiable by software.

At the end of a successful double-word program operation, the flash module indicates to the TDM the address of the programmed double word. The TDM determines whether the programmed data was written to the correct TDR diary location and then unlocks the

blocks for erase operation for the corresponding TDR. The TDM ensures that this erase operation on unlocked blocks is allowed until the completion of the erase sequence or until next destructive reset.

When attempting to update the TDM diary and erase flash blocks, set MCR[PECIE] in the Embedded Flash Memory registers to enable interrupt notification for program and erase operations. Then clear MCR[PECIE] to service the interrupt after each program or erase operation. This sequence must be executed before attempting to update the TDM diary, and then again before attempting to erase the unlocked flash blocks.

NOTE

All blocks associated to a TDR where an erase operation is allowed can be erased all together.

The data programmed in the diary location is flexible, and may include count information or other important customer information.

NOTE

There is no restriction on the type or format of data that can be programmed in the diary, so software must verify whether the diary is full. If a TDR's diary area is full, the tamper detect mechanism can be disabled for that TDR by creating a Tamper Region Override (TO) DCF record with the TDR's corresponding TDR Override Enable (TOE) bit set to '1'. This allows erase operations to be performed to the TDR without any diary entry.

The setting from “override tamper region lock” DCF record ([Diary](#)) has priority over the settings in the TDR_n_LOCK0 – 3 registers. In other words, if a TDR is unlocked for erase by the override tamper region lock, the TDR_n_LOCK_m bits have no effect on the TDR that has been overridden, but the tamper detection remains active for all TDRs not having the override bit set.

Moreover, if a block is locked by the PASS_LOCK_x_PG_n registers or by TDR_n_LOCK_m, the block is locked for erase.

NOTE

The settings in the Tamper Region Override Register have priority over the TDR_n_LOCK_m registers. The PASS module also has priority over the settings done in the TDM. For instance, if blocks in a TDR are unlocked for erase operation via the TDM, but the same blocks are programmed to be locked for erase operation in PASS, the blocks end up being locked for erase operation.

Attempts to erase the flash without first writing the diary will be ignored by the flash. The contents in the flash block will remain unchanged and MCR[PEG] will be asserted, indicating the operation was successful and the contents of the block are properly protected from the erase operation.

68.7.2 Assigning blocks to Tamper Detect Regions (TDR)

Each TDR is defined by a group of 4 x 32-bit DCF records and clients, with each containing bits that map to specific blocks in a flash region. These mappings mirror exactly the LOCK registers bits from the flash module. Hence, the block assignment registers are named as $TDR_n\_LOCK_m$ where n is the number of a tamper region, that is, 0- 5 and m varies from 0-3 as flash contains four Lock registers.

In these DCF records:

- '1' means that the corresponding flash block is assigned to this TDR
- '0' means that the corresponding flash block is not assigned to this TDR (Default)

DCF clients are not visible to software. They can not be read nor written. They can only be initialized by DCF record within the UTest flash block during reset. The value of these registers can be evaluated by reading all the UTest DCF records and searching for records that correspond to these clients.

NOTE

No flash block should be assigned to more than one TDR. Each block should correspond to a single TDR. For example, if a single block is assigned to TDR0 and TDR1, then even if the Diary is programmed for both TDR0 and TDR1, this flash block will never be unlocked for erase operation.

68.7.3 Diary

The diary is a region of the flash memory where records of block erases for each TDR are stored. The format and size of the records are not fixed. The only requirements are that each diary has a maximum size of 2 KB, and the minimum size of any programming operation is 8 bytes to honor OTP restrictions. Therefore, each diary can hold a maximum of 256 64-bit entries.

A flash region block or part of it must be assigned for implementing the diary. The base address for the diary is determined by a value written in a DCF record. The diary can be placed within any flash block in flash array, and history records in it can be read by any core.

NOTE

To maintain the security of the TDR, the block where the diary is placed must be assigned as OTP within the OTP registers.

The base address of the diary is implemented using a DCF Client. The DCF client is only writable once and the default bit state is 1.

The DCF clients can not be read nor written by software. They can only be initialized by DCF record from within UTEST block during reset. The value of these registers can be evaluated by reading all the UTEST DCF records and searching for records that write these DCF clients.

68.7.4 Specifying the diary base address

The diary is a region of flash memory where records of block erases for each TDR are stored. The format and size of these records is defined by the customer. The only hardware constraints are that each diary has a maximum size of 2 KB and the minimum size of any programming operation is 8 bytes. Therefore, each diary region limits the erases of the assigned blocks to 256 erases.

The diary can be placed within any flash block in the flash array. To maintain the security of the TDR, the block where the diary is placed, must be assigned as OTP within the OTP registers, while in customer delivery or OEM production life cycle states.

The base address of the diary is implemented with a 20-bit DCF client that can be initialized by the SSCM during reset. The DCF client is writable once and the default bit state is '1'.

The base address register is not visible to software. It cannot be read nor written. It can only be initialized by DCF record within UTest flash during reset. The value of this register can be evaluated by reading all the UTest flash DCF Records and searching for record that writes this register.

Each TDR is assigned a 2 KB section of the diary. With 6 TDRs, 12 KB of diary space is required.

The diary base address must be at a 16 KB boundary. Therefore the least significant 14 bits of the base address must all be '0'.

The Base address of the diary is represented by:

Diary_Base_Address = 0bxxxx_xxxx_xxxx_xxxx_xx00_0000_0000_0000

The base address of each TDR is shown below:

- TDR0 = Diary_Base_Address + 0b00_0000_0000_0000

- $\text{TDR1} = \text{Diary\_Base\_Address} + 0b00\_1000\_0000\_0000$
- $\text{TDR2} = \text{Diary\_Base\_Address} + 0b01\_0000\_0000\_0000$
- $\text{TDR3} = \text{Diary\_Base\_Address} + 0b01\_1000\_0000\_0000$
- $\text{TDR4} = \text{Diary\_Base\_Address} + 0b10\_0000\_0000\_0000$
- $\text{TDR5} = \text{Diary\_Base\_Address} + 0b10\_1000\_0000\_0000$
- End of the diary = $\text{Diary\_Base\_Address} + 0b11\_0000\_0000\_0000$

68.7.5 TDR lock status

The status of whether a TDR is locked for erase is maintained by the TDM. This is accomplished by implementing a register with as many bits as the number of TDRs. The default state of these bits is 1, which is defined as "TDR Erase Blocked". See [TDR Status Register \(TDM_TDRSR\)](#) for further details about this register.

When the flash memory module has completed a programming operation, the address of the location programmed is output from the flash module. The TDM in turn compares that address to determine if it falls within the range of any TDR diary regions. If it is within any TDR, the corresponding status bit is set to 0 meaning "TDR Erase Enabled".

These status bits are all reset to 1 when the flash module signals an erase complete by asserting the PEC (Program Erase Complete) signal.

The state of the 6 status bits can be read through the TDRSR. See [TDR Status Register \(TDM_TDRSR\)](#). The address of the last successful programming operation provided from the flash memory to the TDM can be read through the LFPAR. See [Last Flash Programmed Address Register \(TDM_LFPAR\)](#).

68.7.6 TDR override

Before a block protected by a tamper detect region can be erased, a record must be written to the associated diary. Software must search through the diary to find the end of the records, where the next available diary location that can be written. After a maximum of 256 records are programmed to the diary, a TDR diary is full. Full means that at least 1 bit has been programmed to '0' in each 64 bit double word within the 2 KB diary.

If software determines that a particular TDR diary is full it may override the diary to allow erasing to continue or it may choose to stop further erases and to flag an error.

Diary Override bits are implemented with a DCF client consisting of 6 bits, one for each TDR. See the device configuration information at the beginning of this chapter for TDM DCF client details.

The Diary Override bits are by default set to '0'; which means the diary must be written before an erase can occur. The DCF client can be written many times but the bits can only be written to '1'. Writes to '0' are ignored. In this way, by default, all Tamper Detect Regions are active. The Tamper detect region diaries can be individually overridden by writing a DCF record that sets that corresponding bit to '1'. In this mode, the TDR status bit is permanently set to 0 to allow flash erase.

The application can also override the diary by loading a service key to the `STO_KEYn` program-visible register, where *n* denotes the tamper detect region. By loading the value `0x55AA5A5A` into `STO_KEYn` register, the corresponding tamper detect region is overridden. This software override mechanism is disabled if the `STO_DISn` field of the `STO_DIS` DCF Client is set.

68.7.7 One time programmable (OTP)

68.7.7.1 Overview

Any flash block within the flash array can be assigned, at any time, to be OTP. Once a flash block is assigned OTP, it cannot be changed back.

OTP means that flash erase of the entire block is disabled and only 64-bit double words that are already erased (that is, flash content is `0xFFFF_FFFF_FFFF_FFFF`) can be programmed. Over-programming is not possible.

Flash blocks are assigned as OTP by writing a DCF record. The block will become OTP after the next functional reset.

Appendix A

Release Notes

A.1 General changes throughout and Preface

- In eDMA chapter, removed "Lockstep" section

A.2 Introduction changes

- No substantial content changes

A.3 Signal Description changes

- No substantial content changes

A.4 Memory Map, Interrupts, and DMA Requests changes

- In [Table 4-3](#) of [Memory map](#)
 - In footnote associated with row for Unique Chip Identifier (UID), changed Freescale to NXP
 - In row for Life cycle slot 0 and associated footnote, changed "Freescale Production" to "MCU Production"
- In [Table 4-4](#) of [Interrupt vector assignments](#)
 - Changed every *ETPU* reference to *eTPU*
 - In rows for vectors 68-99, changed eTPU_CISR_A to eTPU_GCR_CISR_ENG1
 - In rows for vectors 243-274, changed eTPU_CISR_B to eTPU_GCR_CISR_ENG2
 - In rows for vectors 370-393 and 493-500, changed eTPU_CISR_C to eTPU_GCR_CISR_ENG1

A.5 Device Configuration Format (DCF) Records changes

- Added new note in multiple sections: "Be careful when programming the UTEST Miscellaneous, XOSC ANA, and VREG Enable DCF clients. DCF records are processed sequentially in the order they are read from flash memory. DCF records that control actions such as enabling/disabling voltage regulators or detectors should not be programmed more than once with different values."
 - In [Table 5-6](#) of [DCF clients](#) added as new footnote referenced in "DCF client special strategy" column for UTEST Miscellaneous, UTEST XOSC ANA, and VREG Enable clients
 - Added as Note in descriptions of:
 - [UTEST Miscellaneous DCF client](#)
 - [UTEST Analog External Oscillator \(XOSC ANA\) DCF client](#)
 - [UTEST PMC Voltage Regulator Enable \(VREG ENB\) DCF client](#)
- In [DCF clients](#) section, in first body row of [Table 5-6](#) about STCU2 clients, removed reference to "32-bit PRPG"

A.6 e200z7 Core Complex Overview changes

- No substantial content changes

A.7 PCU changes

- No substantial content changes

A.8 System Integration Unit (SIU) changes

- No substantial content changes

A.9 Crossbar Switch (XBAR) changes

A.9.1 Chip-specific XBAR information changes

- No substantial content changes

A.9.2 Crossbar switch module changes

- No substantial content changes

A.10 Crossbar Integrity Checker (XBIC) changes

A.10.1 Chip-specific XBIC information changes

- No substantial content changes

A.10.2 XBIC module changes

- No substantial content changes

A.11 Peripheral Bridge (AIPS/PBRIDGE) changes

A.11.1 Chip-specific AIPS/PBRIDGE information changes

- No substantial content changes

A.11.2 AIPS module changes

- No substantial content changes

A.12 Platform Configuration Module (PCM) changes

- No substantial content changes

A.13 SEMA4 changes

- No substantial content changes

A.14 Memory Protection Unit (MPU) changes

A.14.1 Chip-specific MPU information changes

- No substantial content changes

A.14.2 MPU module changes

- In [Control/Error Status Register \(MPU_CESR\)](#) description of SPERR field, corrected correlation between each field bit and corresponding slave port

A.15 Interrupt Controller (INTC) changes

A.15.1 Chip-specific INTC information changes

- No substantial content changes

A.15.2 INTC module changes

- No substantial content changes

A.16 Direct Memory Access Controller (eDMA) changes

A.16.1 Chip-specific eDMA information changes

- No substantial content changes

A.16.2 eDMA module changes

- [Error Status Register \(DMA_ES\)](#) : In description, replaced "See the Error Reporting and Handling section" with "See [Fault reporting and handling](#)."

A.17 Error Injection Module (EIM) changes

A.17.1 Chip-specific EIM information changes

- No substantial content changes

A.17.2 EIM module changes

- No substantial content changes

A.18 Error Reporting Module (ERM) changes

A.18.1 Chip-specific ERM information changes

- No substantial content changes

A.18.2 ERM module changes

- Added section: [Initialization](#)

A.19 Clocking changes

- No substantial content changes

A.20 Progressive Clock Switch (PCS) changes

- No substantial content changes

A.21 Dual PLL Digital Interface (PLLDIG) changes

- No substantial content changes

A.22 CMU changes

- [CMU Interrupt Status Register \(CMU_ISR\)](#)
 - Updated Note in OLRI field description.
- [CMU Low Frequency Reference Register CLKMN1 \(CMU_LFREFR\)](#)
 - Added the note "For correct synchronization of CMU_LFREFR write..." to the register description.
- [CMU High Frequency Reference Register CLKMN1 \(CMU_HFREFR\)](#)
 - Updated the note "For correct synchronization of CMU_HFREFR write..." to the register description.
- [CLKMN1 supervisor](#)
 - Updated NOTE "An example of determining the HFREF_{Actual}..."
- In [CMU Interrupt Status Register \(CMU_ISR\)](#)
 - Updated NOTE "All flags in the CMU_ISR are set asynchronously. This..."
- In [CMU Interrupt Status Register \(CMU_ISR\)](#)
 - Updated NOTE "Before entering low-power stop modes,..."
- [CMU Control Status Register \(CMU_CSR\)](#)
 - Added note to CSR[SFM] field description, "MSR[MD] must be written before enabling..."

A.23 Embedded Memories changes

- No substantial content changes

A.24 RAM Controller (PRAMC) changes

A.24.1 Chip-specific PRAMC information changes

- No substantial content changes

A.24.2 PRAMC module changes

- No substantial content changes

A.25 Flash Memory Controller (PFLASH) changes

A.25.1 Chip-specific PFLASH information changes

- No substantial content changes

A.25.2 Flash memory controller module changes

- [Flash memory controller memory map](#) :
 - Typo fixed: "If application software functionality includes invalidating a port read buffer, t recommended method..." changed to, "If application software functionality includes invalidating a port read buffer, the recommended method..."
- [ECC on data flash accesses](#) :
 - Content significantly rearranged.

A.26 Embedded Flash Memory (C55FMC) changes

A.26.1 Chip-specific C55FMC information changes

- No substantial content changes

A.26.2 c55fmc module changes

- No substantial content changes

A.27 External Bus Interface (EBI) changes

A.27.1 Chip-specific EBI information changes

- No substantial content changes

A.27.2 EBI module changes

- No substantial content changes

A.28 Analog Modules Overview changes

- No substantial content changes

A.29 Sigma Delta Analog-to-Digital Converter (SDADC) Digital Interface changes

A.29.1 Chip-specific SDADC information changes

- No substantial content changes

A.29.2 SDADC module changes

- No substantial content changes

A.30 Enhanced Queued Analog-to-Digital Converter (EQADC) changes

A.30.1 Chip-specific EQADC information changes

- No substantial content changes

A.30.2 EQADC module changes

- In [Conversion command format for the standard configuration](#) removed stray line between bits 22 and 23 in [Figure 30-117](#)

A.31 Decimation Filter (DECFILTER) changes

A.31.1 Chip-specific DECFILTER information changes

- No substantial content changes

A.31.2 DECFILTER module changes

- No substantial content changes

A.32 Temperature Sensor changes

A.32.1 Chip-specific Temperature Sensor information changes

- No substantial content changes

A.32.2 Temperature Sensor changes

- No substantial content changes

A.33 System Timer Module (STM) changes

A.33.1 Chip-specific STM information changes

- No substantial content changes

A.33.2 STM module changes

- No substantial content changes

A.34 Software Watchdog Timer (SWT) changes

A.34.1 Chip-specific SWT information changes

- In [SWT time-out events and invalid accesses: configuring reset and interrupt requests](#)
 - Added information about invalid accesses and SWT_CR[RIA], including [Table 34-3](#)
 - Reorganized section to accommodate new information

A.34.2 SWT module changes

- In "Memory Map and Registers"
 - In introductory text, incorporated minor edits and reorganized some paragraph text as list
 - Incorporated minor edits in each register description
- In [SWT Control Register \(SWT_CR\)](#) description of [ITR](#) field
 - In Note, changed "reset requests" to "reset requests and interrupt requests"
- In [SWT Control Register \(SWT_CR\)](#) description of [RIA](#) field
 - Added a Note: "For a description of how this chip implements SWT reset requests resulting from SWT invalid accesses, see the chip-specific SWT information."
 - Edited description of field value 1
- In [Window mode](#) edited and reorganized entire section

A.35 Periodic Interrupt Timer (PIT-RTI) changes

A.35.1 Chip-specific PIT information changes

- No substantial content changes

A.35.2 PIT module changes

- No substantial content changes

A.36 Enhanced Modular IO Subsystem (eMIOS) changes

A.36.1 Chip-specific eMIOS information changes

- No substantial content changes

A.36.2 eMIOS module changes

- No substantial content changes

A.37 Enhanced Time Processing Unit (eTPU) changes

A.37.1 Chip-specific eTPU information changes

- No substantial content changes

A.37.2 eTPU module changes

- Editorial changes (including the addition of cross-references for improved usability).
- Throughout the chapter, revised register names so that they are always consistent with the official register names as specified in these tables:
 - "eTPU_SCR memory map" in [System Configuration memory map and registers](#)
 - "eTPU_TBR memory map" in [Timebase memory map and registers](#)
 - "eTPU_ER memory map" in [Engine memory map and registers](#)
 - "eTPU_LMR memory map" in [Latency Monitor memory map and registers](#)
 - "eTPU_MESR memory map" in [Memory Error Support memory map and registers](#)
 - "eTPU_GCR memory map" in [Global Channel memory map and registers](#)
 - "eTPU_CCCR memory map" in [Channel Configuration Control memory map and registers](#)
 - "eTPU_SSIR memory map" in [Shared Subsystem Integration memory map and registers](#)
- So, for example, the chapter now consistently uses:
 - eTPU_MESR_CERDR instead of ETPUCERD
 - eTPU_GCR_CDTRER_ENG1 and eTPU_GCR_CDTRER_ENG2 instead of ETPUCDTRER
 - eTPU_TBR_STAC_ENG1 and eTPU_TBR_STAC_ENG2 instead of ETPUREDCE
- In [eTPU Channel n Configuration Register for Engine 1 \(eTPU_CCCR_CnCR_ENG1\)](#), revised the CIE field description.
- In [eTPU Channel n Configuration Register for Engine 1 \(eTPU_CCCR_CnCR_ENG1\)](#), in the DTRE field description, changed "This bit is mirrored from [eTPU Channel Data Transfer Request Enable Register for Engine 2 \(eTPU_GCR_CDTRER_ENG2\)](#)" to "This bit is mirrored from [eTPU Channel Data Transfer Request Enable Register for Engine 1 \(eTPU_GCR_CDTRER_ENG1\)](#)."
- In [eTPU Channel n Configuration Register for Engine 2 \(eTPU_CCCR_CnCR_ENG2\)](#), in the T4DIS field description, added "For more details, see [Channel timing modes](#)."
- In [Table 37-344](#), deleted the footnote "See the eTPU integration guide..."
- In [Latency Monitor](#), in the last bulleted list, changed "after each latency measurement, the value measured is compared with the MAXLAT value in the register ETPULMnMR; and if MAXLAT is not zero and the new measured value is greater." to "After each latency measurement, the value measured is compared with the LATENCY value in [eTPU Latency Monitor Measure Register for Engine 1 \(eTPU_LMR_LMnMR_ENG1\)](#) and [eTPU Latency Monitor Measure Register for Engine 2 \(eTPU_LMR_LMnMR_ENG2\)](#). If the new measured value is greater, it replaces the old value in [eTPU Latency Monitor Measure Register for Engine 1 \(eTPU_LMR_LMnMR_ENG1\)](#) and [eTPU Latency Monitor Measure Register for Engine 2 \(eTPU_LMR_LMnMR_ENG2\)](#)."
- In [Using Worst-Case Latency estimates to evaluate performance](#), changed "Refer to Freescale Programming Note" to "See NXP Programming Note".
- In [Execution and channel timing](#), deleted "For timings of eTPU block internal interface signals, consult the eTPU Integration Guide. For timings internal to the eTPU, refer to the eTPU Creation Guide."
- Editorial changes in [PRSS - Pin Request Service Sample](#).
- In [Unsigned Fractional Multiplication \(fmultu\)](#), added the footnote "24-bit fractional signed/unsigned multiplication is equivalent to mults/multu." to the second sentence.
- In [TDLA/B - Transition Detect Latches](#), added the footnote "In TPU3, when TCR1 was counting at maximum rate of eTPU clock divided by 2, the next value was captured." to the fourth paragraph.

A.38 Reaction Module (REACM2) changes

A.38.1 Chip-specific REACM2 information changes

- No substantial content changes

A.38.2 REACM2 module changes

- No substantial content changes

A.39 Input Glitch Filter (IGF) changes

A.39.1 Chip-specific IGF information changes

- No substantial content changes

A.39.2 IGF module changes

- No substantial content changes

A.40 FlexCAN (CAN) changes

A.40.1 Chip-specific FlexCAN information changes

- No substantial content changes

A.40.2 FlexCAN module changes

- No substantial content changes

A.41 Modular CAN (M_CAN) changes

A.41.1 Chip-specific M_CAN information changes

- No substantial content changes

A.41.1.1 M_CAN Subsystem changes

- No substantial content changes

A.41.2 M_CAN module changes

- In second Note of [Protocol Status Register \(M_CAN_PSR\)](#) description, revised text and reorganized into list
 - from: "The Bus_Off recovery sequence (see CAN Specification Rev. 2.0 or ISO11898-1) cannot be shortened by setting or resetting CCCR[INIT]. If the device goes Bus_Off, it will set CCCR[INIT] of its own accord, stopping all bus activities. Once CCCR[INIT] has been cleared by the CPU, the device will then wait for 129 occurrences of Bus Idle (129 * 11 consecutive recessive bits) before resuming normal operation. At the end of the Bus_Off recovery sequence, the Error Management Counters will be reset. During the waiting time after the resetting of CCCR[INIT], each time a sequence of 11 recessive bits has been monitored, a Bit0Error code is written to PSR.LEC, enabling the CPU to readily check up whether the CAN bus is stuck at dominant or continuously disturbed and to monitor the Bus_Off recovery sequence. ECR.REC is used to count these sequences."
 - to: "The Bus_Off recovery sequence (see CAN Specification Rev. 2.0 or ISO11898-1) cannot be shortened by setting or resetting CCCR[INIT].
 1. If the device goes Bus_Off, it sets CCCR[INIT] and PSR[BO], stopping all bus activities.
 2. After the user clears CCCR[INIT], the user must then wait for 129 occurrences of Bus Idle (129 * 11 consecutive recessive bits) before normal CAN operation can resume.
 - During the waiting period, each time the device monitors a sequence of 11 recessive bits, a Bit0Error code (101b) is written to PSR[LEC].
 - These code writes enable the user to check readily whether the CAN bus is stuck at dominant or continuously disturbed and to monitor the Bus_Off recovery sequence.
 - ECR[REC] is used to count these sequences.
 3. After the observation of the 129 occurrences of Bus Idle, the device clears PSR[BO]."

A.42 Deserial Serial Peripheral Interface (DSPI) changes

A.42.1 Chip-specific DSPI information changes

- No substantial content changes

A.42.2 DSPI module changes

- In section, [Memory Map/Register Definition](#) : Added note
- In [Serial Peripheral Interface \(SPI\) configuration](#) removed redundant list item for "Transmit First In First Out (TX FIFO) buffering mechanism"
- In [Transmit FIFO Fill Interrupt or DMA Request](#) added text to Note: "Configure the DMA to fill only one FIFO location per transfer."
- In [Receive FIFO Drain Interrupt or DMA Request](#) added text: "Configure the DMA to drain only one FIFO location per transfer."

- In [PUSH TX FIFO Register In Master Mode \(DSPI_PUSHR\)](#) description, changed second sentence from "Only 16-bits can be written into TXDATA field" to "User must write 16-bits data into TXDATA field."
- Updated bit field descriptions for [SR\[TFFF, RFDF\]](#)
- In [Status Register \(DSPI_SR\)](#), changed access of [Reserved](#) field from read-only to "write 1 to clear"
- In [Status Register \(DSPI_SR\)](#), edited field description of:
 - [RFDF](#)
 - [TFFF](#)
- In [Memory Map/Register Definition](#) Note, edited text and reorganized items as a list

A.43 Zipwire changes

A.43.1 Chip-specific Zipwire information changes

- No substantial content changes

A.43.2 Zipwire module changes

- No substantial content changes

A.44 Serial Interprocessor Interface (SIPI) changes

A.44.1 Chip-specific SIPI information changes

- No substantial content changes

A.44.2 SIPI module changes

- In section [DMA programming sequence](#)
 - List item 5 changed from "If SIPI has received..., move to Step 8." to "When SIPI receives read...moves to step 8."
 - List item 6 changed from "As soon as ipd_request is asserted, AHB transfer...SIPI registers and RAM." to "When SIPI initiates DMA request,...SIPI and RAM."

A.45 LVDS Fast Asynchronous Serial Transmission (LFAST) – Interprocessor Communications changes

A.45.1 Chip-specific LFAST information changes

- No substantial content changes

A.45.2 LFAST module changes

- Added bit description text "*This field can only be written when MCR[DRFEN] = 1.*" in [CLKTST](#), [LPON](#), [PNGREQ](#) and [USNDRQ](#) bits.
- In [LFAST PLL Control Register \(LFAST_PLLCR\)](#) register: updated "LPCFG" bit as Reserved now.

A.46 Fast Ethernet Controller (FEC) changes

A.46.1 Chip-specific FEC information changes

- No substantial content changes

A.46.2 FEC module changes

- No substantial content changes

A.47 Peripheral Sensor Interface (PSI5) changes

A.47.1 Chip-specific PSI5 information changes

- In [PSI5 GTM interface](#) clarified that this chapter refers to eTPU_C as the general timer module (GTM)

A.47.2 PSI5 module changes

- In "Memory map and register description," updated bitfield descriptions for PSI5_CH0_DCR[DMA_PM_DS_WM] and [DMA_EN_SF].

A.48 SENT Receiver (SRX) changes

A.48.1 Chip-specific SRX information changes

- No substantial content changes

A.48.2 SRX/SENT module changes

- No substantial content changes

A.49 Enhanced Serial Communication Interface (eSCI) changes

A.49.1 Chip-specific eSCI information changes

- No substantial content changes

A.49.2 eSCI module changes

- In the [Control Register 1 \(eSCI_CR1\)Table 49-7](#), corrected table header typo "RSCR" to "RSRC".

A.50 Resets changes

- No substantial content changes

A.51 System Boot changes

- In [Features](#) changed Freescale to NXP in second list item
- In [Reset Configuration Half Word \(RCHW\)](#) description of VLE, changed Freescale to NXP

A.52 System Status and Configuration Module (SSCM) changes

A.52.1 Chip-specific SSCM information changes

- In [Life Cycle \(LC\) slots](#) changed "Freescale Production" to "MCU Production" in final column of first body row in [Table 52-1](#)

A.52.2 SSCM module changes

- "Freescale production" is changed to "MCU Production" in following topics:
 - In bitfield value description of 110 in [LC](#).
 - [Life Cycle](#).
 - [Table 52-14](#)
- In [Figure 52-6](#) updated "Freescale production" to "MCU Production".
- In [SSCM \(SSCM\)](#) added a note "Reserved IPS register access is prohibited"

A.53 Power Management changes

- No substantial content changes

A.54 PMC changes

- No substantial content changes

A.55 JTAG Controller (JTAGC) changes

A.55.1 Chip-specific JTAGC information changes

- No substantial content changes

A.55.2 JTAGC module changes

- In [Table 55-2](#) of [External signal description](#) :
 - Moved content from Pull column to Reset State column and removed Pull column
 - Updated "Up" to "Weak pullup"
 - Updated "Down" to "Weak pulldown"

A.56 CJTAG changes

- No substantial content changes

A.57 JDC changes

- No substantial content changes

A.58 Nexus Development Interface (NDI) changes

- No substantial content changes

A.59 Nexus Port Controller (NPC) changes

A.59.1 Chip-specific NPC information changes

- No substantial content changes

A.59.2 NPC module changes

- Minor editorial change in [IEEE 1149.1-2001 \(JTAG\) TAP](#).

A.60 Nexus eTPU Development Interface (NDEDI) changes

A.60.1 Chip-specific NDEDI information changes

- No substantial content changes

A.60.2 NDEDI module changes

- No substantial content changes

A.61 DTS changes

- No substantial content changes

A.62 Fault Collection and Control Unit (FCCU) changes

A.62.1 Chip-specific FCCU information changes

- No substantial content changes

A.62.2 FCCU module changes

- In [More about the EOUT interface](#), changed *<i>NA</i> (not applicable) appears in the following table* to *NA (not applicable) appears in the following table*.
- In the [Bistable Fault-Output \(EOUT\) Mode Signaling Enable \(FCCU_EOUT_SIG_ENn\)](#) register description, rewrote the register's function description and the [EOUTENx](#) field's function description.
- In [Signals](#), revised the function description for the FIF signal.

- Throughout the chapter, changed *non-critical* to *noncritical*.
- In each of the following register descriptions, changed the format of the table in the register function summary and reworded the sentence in the field function summary that refers to the table:
 - [Noncritical Fault Configuration \(FCCU_NCF_CFGn\)](#)
 - [Noncritical Fault Status \(FCCU_NCF_Sn\)](#)
 - [Noncritical Fault Enable \(FCCU_NCF_En\)](#)
 - [Noncritical Fault Timeout Enable \(FCCU_NCF_TOEn\)](#)
 - [IRQ Alarm Enable \(FCCU_IRQ_ALARM_ENn\)](#)
 - [Bistable Fault-Output \(EOUT\) Mode Signaling Enable \(FCCU_EOUT_SIG_ENn\)](#) (no sentence rewording)
- In [Introduction](#), changed *One of the error out pins is high* to *One of the fault-output (EOUT) signals is high*.
- In the [Signals](#) table, deleted the rows for:
 - WKPU interface (divider row)
 - NMIWKPU
 - WAKEWKPU

A.63 Self-Test Control Unit (STCU2) changes

A.63.1 Chip-specific STCU2 information changes

- In [STCU register default values](#) table, removed rows about STCU2_LB_PRPGLn and STCU2_LB_PRPGHn registers

A.63.2 STCU2 module changes

- Added the following sections to clarify terminology:
 - [BISTs and BIST partitions](#)
 - [BIST sequences](#)
- [STCU2 Watchdog Granularity \(STCU2_WDG\)](#) :
 - Throughout: Edited for clarity
 - Note added to WDG[WDGEOC] field description: "If this register is programmed while STCU2_RUN[RUN]=1, the new STCU2_WDG value, programmed after STCU2_RUN, does not take effect."
- [STCU2 LBIST Control \(STCU2_LB_CTRLn\)](#) :
 - Changed 4-bit, RO field (bits 8 - 11) to 3-bit field
 - Changed access level of reserved RO bit 11 to RW
 - Replaced PRPGEN field with R/W reserved field
- [STCU2 LBIST PRPG Low Register \(STCU2_LB_PRPGLn\)](#) :
 - Register removed
- [STCU2 LBIST PRPG High Register \(STCU2_LB_PRPGHn\)](#) :
 - Register removed
- [Offline self-test sequence](#) :
 - Removed: "Optionally, also the STCU2_LB_PRPGL/H can be programmed in order to load a specific seed."
- In the [STCU2 Off-Line MBIST Status Medium Register \(STCU2_MBSM\)](#) register description, in the note in each of the field descriptions, changed *STCU2_MBEL* to *STCU2_MBEH*.
- In the [STCU2 Off-Line MBIST Status High Register \(STCU2_MBSH\)](#) register description, in the note in each of the field descriptions, changed *STCU2_MBEL* to *STCU2_MBEH*.

- [Online self test sequence](#) :
 - Numerous editorial changes
 - Steps changed to a numbered sequence instead of a bulleted list
 - STCU2 Register names referenced in this section are no longer prefixed with "STCU2_".
- In the [STCU2 Configuration Register \(STCU2_CFG\)](#) register description, changed *FSL production* to *MCU production*.

A.64 Cyclic Redundancy Check (CRC) Unit changes

A.64.1 Chip-specific CRC information changes

- No substantial content changes

A.64.2 CRC module changes

- In [Configuration Register \(CRC_CFGn\)](#), updated reset value footnote for POLYG field.

A.65 Security Overview changes

- Added section: [Flash memory read, write, and erase protection](#)

A.66 Cryptographic Services Engine (CSE) changes

A.66.1 Chip-specific CSE information changes

- Added section: [CSE clock: minimum frequency](#)
- Changed content of [CSE command-processing completion before Stop mode entry](#)
 - from: "To ensure proper CSE operation: Before the chip enters Stop mode, CSE_SR[BSY] must be 0."
 - to: "To ensure proper CSE operation: Before setting SIU_HLT1[CSE] to 1, ensure CSE_SR[BSY] is 0."

A.66.2 CSE module changes

- No substantial content changes

A.67 Password and Device Security Module (PASS) changes

A.67.1 Chip-specific PASS information changes

- No substantial content changes

A.67.2 PASS module changes

- In register bit field: [LIFE](#) :
 - Changed 'Freescale Production' to 'MCU Production'
- Editorial change: Added cross ref in [Censorship](#)

A.68 Tamper Detection Module (TDM) changes

A.68.1 Chip-specific TDM information changes

- No substantial content changes

A.68.2 TDM module changes

- In section [Flash erase counter and tamper detection](#)
 - Removed text "Alternatively,...the diary length for that TDR." from second note in this section.



How to Reach Us:**Home Page:**nxp.com**Web Support:**nxp.com/support

Information in this document is provided solely to enable system and software implementers to use NXP products. There are no express or implied copyright licenses granted hereunder to design or fabricate any integrated circuits based on the information in this document. NXP reserves the right to make changes without further notice to any products herein.

NXP makes no warranty, representation, or guarantee regarding the suitability of its products for any particular purpose, nor does NXP assume any liability arising out of the application or use of any product or circuit, and specifically disclaims any and all liability, including without limitation consequential or incidental damages. "Typical" parameters that may be provided in NXP data sheets and/or specifications can and do vary in different applications, and actual performance may vary over time. All operating parameters, including "typicals," must be validated for each customer application by customer's technical experts. NXP does not convey any license under its patent rights nor the rights of others. NXP sells products pursuant to standard terms and conditions of sale, which can be found at the following address: nxp.com/SalesTermsandConditions.

NXP, the NXP logo, Freescale, the Freescale logo, SafeAssure, the SafeAssure logo, and MXC are trademarks of NXP B.V. All other product or service names are the property of their respective owners. The Power Architecture and Power.org word marks and the Power and Power.org logos and related marks are trademarks and service marks licensed by Power.org.

© 2013–2016 NXP B.V.